Tomus 7. Fasciculus 2.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1985
Curat: Universitas Szegediensis de Attila József nominata

7. kötet 2. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

Szeged, 1985. július
A Szegedi József Attila Tudományegyetem gondozásában

On the weak equivalence of Elgot's flow-chart schemata

by Z. ÉSIK

Notions and notations

Algebraic theories were originally introduced in [10]. An equational presentation of algebraic theories can be found in [1]. Following this latter work, by an algebraic theory we shall mean a many-sorted algebra $T=(T(n,p); \cdot, \langle\ \rangle, \pi_p^i)$ where n, p are non-negative integers; composition, denoted by $\cdot$ or juxtaposition, maps $T(n,p)\times T(p,q)$ into $T(n,q)$; source-tupling associates a unique element $\langle f_1, ..., f_n\rangle \in T(n,p)$ with each family of scalar elements $f_1, ..., f_n \in T(1,p)$; finally, there is an injection $\pi_p^i \in T(1,p)$ for each i and p such that $i \in [p]$ ($[p]=\{1, ..., p\}$). Furthermore, the following identities have to be satisfied by T:

(A_1) $f(gh)=(fg)h$ if $f\in T(m,n)$, $g\in T(n,p)$, $h\in T(p,q)$,

(A_2) $f\langle \pi_p^1, ..., \pi_p^p\rangle = f$ if $f\in T(n,p)$,

(A_3) $\pi_n^i\langle f_1, ..., f_n\rangle = f_i$ if $f_1, ..., f_n \in T(1,p)$,

(A_4) $\langle \pi_n^1 f, ..., \pi_n^n f\rangle = f$ if $f\in T(n,p)$.

Although identities (A_1), ..., (A_4) above are sufficient to characterize algebraic theories, in order to have identity $\langle f\rangle = f$ if $f\in T(1,p)$ we require identity

(A_5) $\langle \pi_1^1\rangle = \pi_1^1$ [1].

In case of $n=0$, (A_4) means that $T(0,p)$ is a one-element set, its unique element will be denoted by 0_p. It follows from the axioms that elements $1_n=\langle \pi_n^1, ..., \pi_n^n\rangle$ are identities with respect to composition. Therefore, an algebraic theory can be viewed as a small category. According to this analogy, we shall often write $f: n\to p\in T$ instead of $f\in T(n,p)$.

Pairing, denoted also by $\langle\ \rangle$, and separated sum, which will be denoted by $+$, are frequently used derived operations in algebraic theories. As regards the defini-

[1] In spite of the fact that identity $\langle \pi_1^1\rangle = \pi_1^1$ is used many times by several authors, it is usually not explicited stately. This is the case in [6] and [7], too.

tion of these derived operations cf. [3]. Given a mapping $\varrho\colon [n]\to[p]$, there is a corresponding base element $\varrho\colon n\to p\in T$. It is defined by $\varrho=\langle\pi_p^{\varrho(1)}, \ldots, \pi_p^{\varrho(n)}\rangle$. If the mapping ϱ is surjective then the corresponding base element is also called surjective. Injective and bijective base elements are similarly defined. If $\varrho\colon [n]\to[n]$ is bijective then ϱ^{-1} denotes the inverse of ϱ.

Iteration theories were introduced in [2]. They were called generalized iterative theories in [6] and [7].

An iteration theory is an algebraic theory equipped with a new operation, called iteration and usually denoted by †. In an iteration theory $I=(I(n,p);\ \cdot, \langle\ \rangle, \pi_p^i, {}^\dagger)$ iteration maps $I(n, n+p)$ into $I(n,p)$. According to [6], iteration theories can be characterized by the following identities:

(B_1) $(0_n+f)^\dagger = f$ if $f\colon n\to p\in I$,

(B_2) $(f+0_q)^\dagger = f^\dagger+0_q$ if $f\colon n\to n+p\in I$,

(B_3) $\langle f, g\rangle^\dagger = \langle h^\dagger, (g\varrho)^\dagger\langle h^\dagger, 1_p\rangle\rangle$ where $f\colon n\to n+m+p\in I$,

$g\colon m\to n+m+p\in I$, $\varrho = \langle 0_m+1_n, 1_m+0_n\rangle+1_p$,

$h = f\langle 1_n+0_p, (g\varrho)^\dagger, 0_n+1_p\rangle$,

(B_4) $\langle\pi_m^1\varrho g(\varrho_1+1_p), \ldots, \pi_m^m\varrho g(\varrho_m+1_p)\rangle^\dagger = \varrho(g(\varrho+1_p))^\dagger$ if

$g\colon n\to m+p\in I$, and $\varrho\colon m\to n\in I$, $\varrho_1, \ldots, \varrho_m\colon m\to m\in I$

are base with $\varrho_1\varrho=\ldots=\varrho_m\varrho=\varrho$ and ϱ is surjective,

(B_5) $f\langle f^\dagger, 1_p\rangle = f^\dagger$ if $f\colon n\to n+p\in I$.

(B_5) is called Elgot's fixed-point equation. It was shown in [8] that (B_5) is not independent from the other defining identities of iteration theories. Iteration theories are natural generalizations of iterative theories (cf. [3]) and rational algebraic theories (cf. [13]).

Given a ranked alphabet Σ — i.e. $\Sigma=\cup(\Sigma_n|n=0, 1, \ldots)$ with $\Sigma_n\cap\Sigma_m=\emptyset$ if $n\neq m$, and a fixed countable set of variables $X=\{x_1, x_2, \ldots\}$, the iteration theory of all (partial infinite) Σ-trees on X play an important role in the fixed-point theory of program schemes. Denote by N the set of natural numbers $\{1, 2, \ldots\}$ and by X_n the set of the first n variables $\{x_1, x_2, \ldots, x_n\}$ for each $n\in N$. Furthermore, denote by A^* the set of all strings over a set A. Then, according to [9], the set of n-ary Σ-trees is the set $T_\Sigma^\infty(X_n)$ consisting of all partial functions $f\colon N^*\Sigma \dashrightarrow \cup X_n$ satisfying the following condition:

if $f(wi)$ is defined where $w\in N^*$ and $i\in N$ then also $f(w)$ is defined, and there is an integer $m(\geqq i)$ with $f(w)\in\Sigma_m$.

The n-ary Σ-trees give rise to an iteration theory $T_\Sigma^\infty=(T_\Sigma^\infty(n,p);\ \cdot, \langle\ \rangle, \pi_p^i, {}^\dagger)$, where $T_\Sigma^\infty(n,p)=T_\Sigma^\infty(X_p)^n$ $(n, p\geqq 0)$, composition is defined by tree substitution, source-tupling is the tupling of trees, injection π_p^i is the variable x_i considered to be a p-ary tree, and iteration is defined in the following way: let $f=\langle f_1, \ldots, f_n\rangle\colon n\to n+p\in T_\Sigma^\infty$ and $g=\langle g_1, \ldots, g_n\rangle\colon n\to p\in T_\Sigma^\infty$. Then $f^\dagger=g$ holds provided that

[2] $T_\Sigma^\infty(X_n)$ is denoted by $CT_\Sigma(X_n)$ in [9].

for any $i\in[n]$ and $w\in N^*$, $w\in\operatorname{dom} g_i$ if and only if there exist $r(\geqq 0)$, $i_0(=i)$, $i_1, \ldots, i_r\in[n]$ and $w_j\in\operatorname{dom} f_{i_j}$ $(j=0, \ldots, r)$ such that $f_{i_0}(w_0)=x_{i_1}, \ldots, f_{i_{r-1}}(w_{i_{r-1}})=x_{i_r}, f_{i_r}(w_r)\notin X_n$ and $w=w_0\ldots w_r$. Furthermore, in this case

$$g_i(w)=\begin{cases} f_{i_r}(w_r) & \text{if } f_{i_r}(w_r)\in\Sigma \\ x_j & \text{if } f_{i_r}(w_r)=x_{n+j}. \end{cases}$$

Everywhere in the paper $\perp_{np}$ denotes $(1_n+0_p)^\dagger$. In T_Σ^∞, $\perp_{np}=\langle\perp, \ldots, \perp\rangle$ (n-times), where $\perp$ is the totally undefined nullary tree.

By viewing an n-ary operational symbol $\sigma\in\Sigma_n$ as an n-ary tree, Σ can be embedded into T_Σ^∞ in a natural way. Denote by $R_\Sigma=(R_\Sigma(n,p); \cdot, \langle\,\rangle, \pi_p^i, {}^\dagger)$ the subalgebra generated by Σ in T_Σ^∞. R_Σ is freely generated by Σ in the class of all iteration theories. In more detail, any map $\varphi\colon \Sigma\to I$ into an iteration theory I can be uniquely extended to a homomorphism $\bar{\varphi}\colon R_\Sigma\to I$ provided that φ is a ranked alphabet map, i.e., $\varphi(\Sigma_n)\subseteqq I(1,n)$ $(n\geqq 0)$.

Restricting ourselves to finite Σ-trees we obtain the algebraic theory $T_\Sigma=(T_\Sigma(n,p); \cdot, \langle\,\rangle, \pi_p^i)$. In this theory $T_\Sigma(n,p)=T_\Sigma(X_p)^n$ and $T_\Sigma(X_p)=\{f\in T_\Sigma^\infty(X_p)\mid \operatorname{dom} f \text{ is finite}\}$. Note that T_Σ is a subtheory of R_Σ. Let

$$\overline{T}_\Sigma(X_n)=\{f\in T_\Sigma(X_n)\mid \forall w\in N^*,\ r>0,\ i\in[r],\ f(w)\in\Sigma_r \Rightarrow wi\in\operatorname{dom} f\}.$$

Put $\overline{T}_\Sigma=(\overline{T}_\Sigma(n,p); \cdot, \langle\,\rangle, \pi_p^i)$ where $\overline{T}_\Sigma(n,p)=\overline{T}_\Sigma(X_p)^n$ $(n,p\geqq 0)$. It is well-known that $\overline{T}_\Sigma$ is a subtheory of T_Σ and in fact it is freely generated by Σ in the class of all algebraic theories.

The trees in $\overline{T}_\Sigma(1,p)$ can also be represented as finite strings over the alphabet $\Sigma\cup X_p$. Namely, $\overline{T}_\Sigma(1,p)$ can be viewed as the smallest set satisfying

(i) $X_p, \Sigma_0\subseteqq \overline{T}_\Sigma(1,p)$,

(ii) if $\sigma\in\Sigma_r$, $r>0$, $f_1, \ldots, f_r\in\overline{T}_\Sigma(1,p)$ then $\sigma f_1\ldots f_r\in\overline{T}_\Sigma(1,p)$.

Another interesting iteration theory is the theory $[A]=([A](n,p); \cdot, \langle\,\rangle, \pi_p^i, {}^\dagger)$ on a set A. Here $[A](n,p)$ stands for the set of all partial functions $f\colon A\times[n]\multimap A\times[p]$, $\cdot$ is the composition of partial functions, source-tupling is the source-tupling of partial functions, injection π_p^i is the mapping $a\mapsto(a,i)$ with $A\times[1]$ and A being indentified, finally, if $f\colon A\times[n]\multimap A\times[n+p]$ is a partial function then $f^\dagger$ is the least fixed-point of the mapping $g\mapsto f\langle g, 1_p\rangle$ $(g\colon A\times[n]\to A\times[p])$. Here least means least with respect to the natural ordering of partial functions.

Concerning flow-chart schemata we accept Elgot's definition of flow-chart schemata in [4], with the exception that in order to make iteration to be a totally defined operation rather than a partial one, we allow nodes to be unlabelled in a flow-chart scheme. In this manner, cf. [4], R_Σ becomes the iteration theory of the strong behaviours of finite flow-chart schemata on a ranked alphabet Σ. Therefore, we may treat flow-chart schemata on Σ as elements of R_Σ.

From now on we fix a ranked alphabet Σ with $\Sigma_n=\emptyset$ if $n\neq 1$ and $n\neq 2$, and denote Σ_1 and Σ_2 by Ω and Π, resp. Ω is called the set of action symbols and Π the set of predicate symbols. Furthermore, we shall assume that Π is finite, say $\Pi=\{\pi_1, \ldots, \pi_r\}$. Given a set A, by an interpretatior $\mathscr{I}$ of Σ in A we mean any ranked alphabet map $\mathscr{I}\colon \Sigma\to[A]$ such that $\mathscr{I}(\pi)$ is a total predicate for each $\pi\in\Pi$. That is, if $\mathscr{I}(\pi)(a)=(b,i)$ $(a,b\in A,\ i\in[2])$ then $a=b$, and $\mathscr{I}(\pi)$ is totally defined.

Denote by $\mathscr{I}$ the unique homomorphic extension of $\mathscr{I}$ from R_Σ into $[A]$, as well. We say that $f, g \in R_\Sigma(n,p)$ $(n,p \geqq 0)$ are equivalent under $\mathscr{I}$ provided that $\mathscr{I}(f) = \mathscr{I}(g)$ holds. Moreover, f and g are called weakly equivalent, written $f \equiv g$, if $\mathscr{I}(f) = \mathscr{I}(g)$ holds for every interpretation $\mathscr{I}$ (cf. [4], [11], [12]).

Relation $\equiv$ is a congruence relation of the iteration theory R_Σ. The problem we are going to solve is the presentation of a generating system of this relation. If such a system is found then this system together with the defining identities of iteration theories can be viewed as an axiom system for the weak equivalence of finite flow-chart schemata on Σ.

A generating system of the relation $\equiv$

In the sequel we shall frequently use some consequences of the defining identities of iteration theories. Among these identities there are identities of poor algebraic theories, which will be used without any reference. In the other part of these identities we have identities involving the † operation, and they are listed here:

$(\mathbf{B}_6)$ $(\varrho f(\varrho^{-1}+1_p))^\dagger = \varrho f^\dagger$ if $f\colon n \to n+p$ and $\varrho\colon n \to n$ is bijective,

$(\mathbf{B}_7)$ $\langle f\langle 1_{n+m}+0_{k+p}, h, 0_{n+m+k}+1_p\rangle, g, h\rangle^\dagger = \langle f, g, h\rangle^\dagger$ if $f\colon n \to n+m+k+p$, $g\colon m \to n+m+k+p$, $h\colon k \to n+m+k+p$,

$(\mathbf{B}_8)$ $(f(1_n+g))^\dagger = f^\dagger g$ where $f\colon n \to n+p$, $g\colon p \to q$,

$(\mathbf{B}_9)$ $(1_n+0_1)\langle \bar{a}_1, \ldots, \bar{a}_n, \pi_n^j+0_{1+p}\rangle^\dagger = \langle a_1, \ldots, a_n\rangle^\dagger$ where $a_1, \ldots, a_n\colon 1 \to n+p$, $\bar{a}_k = a_k(1_n+0_1+1_p)$ if $k \neq i$, $\bar{a}_i = a_i(\langle 1_{j-1}+0_{n+2-j}, \pi_{n+1}^{n+1}, 0_j+1_{n-j}+0_1\rangle+1_p)$, $i,j \in [n]$,

$(\mathbf{B}_{10})$ $(1_n+0_m)\langle f(1_n+0_m+1_p), g\rangle^\dagger = f^\dagger$ if $f\colon n \to n+p$, $g\colon m \to n+m+p$,

$(\mathbf{B}_{11})$ $\pi_{n+1}^1\langle \pi_{n+1+p}^2, 0_1+f\rangle^\dagger = \pi_n^1 f^\dagger$ if $f\colon n \to n+p$.

Now we present the system (C) and prove that this system constitutes a generating system of the weak equivalence relation. (C) consists of the following pairs, written as equalities:

$(\mathbf{C}_1)$ $\pi\langle f, f\rangle = f$ if $\pi \in \Pi$, $f\colon 1 \to p \in R_\Sigma$,

$(\mathbf{C}_2)$ $\pi\langle \pi'\langle f_1, f_2\rangle, \pi'\langle f_3, f_4\rangle\rangle = \pi'\langle \pi\langle f_1, f_3\rangle, \pi\langle f_2, f_4\rangle\rangle$ where $\pi, \pi' \in \Pi$, $f_1, \ldots, f_4\colon 1 \to p \in R_\Sigma$,

$(\mathbf{C}_3)$ $\pi\langle \pi\langle f_1, f_2\rangle, \pi\langle f_3, f_4\rangle\rangle = \pi\langle f, f_4\rangle$ where $\pi \in \Pi$, $f_1, \ldots, f_4\colon 1 \to p \in R_\Sigma$,

$(\mathbf{C}_4)$ $f = \perp$ if $f\colon 1 \to 0 \in R_\Sigma$,

$(\mathbf{C}_5)$ $f^\dagger = f\langle \perp_{1p}, 1_p\rangle$ if $f\colon 1 \to 1+p \in T_\Pi$.

Denote by θ the congruence relation induced by (C) in R_Σ. The following statement is immediate by (C)$\subseteq\equiv$.

Lemma 1. $\theta\subseteq\equiv$.

Later on the following statement will be frequently used.

Lemma 2. Let $f\colon n\to n+m+p+q$, $g\colon m\to n+m+p+q$, $h\colon p\to n+m+p+q$ be arbitrary elements in R_Σ. Assume that $(g(\langle 0_m+1_n, 1_m+0_n\rangle+1_{p+q}))^\dagger\theta\bar{g}$ holds where $\bar{g}\colon m\to n+p+q\in R_\Sigma$. Then also $\langle f, g, h\rangle^\dagger\theta\langle f, \bar{g}(1_n+0_m+1_{p+q}), h\rangle^\dagger$.

Proof. First suppose that $p=0$ and let $\varrho=\langle 0_m+1_n, 1_m+0_n\rangle+1_q$. Then $\langle f, g\rangle^\dagger=\langle a^\dagger, (g\varrho)^\dagger\langle a^\dagger, 1_q\rangle\rangle$ follows by (B_3), where $a=f\langle 1_n+0_q, (g\varrho)^\dagger, 0_n+1_q\rangle$. Put $\bar{a}=f\langle 1_n+0_q, \bar{g}, 0_n+1_q\rangle$. As $(g\varrho)^\dagger\theta\bar{g}$, also $a\theta\bar{a}$ and $\langle f, g\rangle^\dagger\theta\langle\bar{a}^\dagger, \bar{g}\langle\bar{a}^\dagger, 1_q\rangle\rangle$. However, $\langle\bar{a}^\dagger, \bar{g}\langle\bar{a}^\dagger, 1_q\rangle\rangle=\langle f, \bar{g}(1_n+0_m+1_q)\rangle^\dagger$ follows by (B_1) and (B_3).

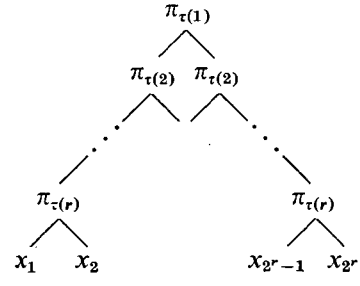
If $p>0$ then define $f_1=f(\alpha+1_q)$, $g_1=g(\alpha+1_q)$ and $h_1=h(\alpha+1_q)$ where $\alpha=1_n+\langle 0_p+1_m, 1_p+0_m\rangle$. Then $(g_1(\langle 0_m+1_{n+p}, 1_m+0_{n+p}\rangle+1_q))^\dagger\theta\bar{g}$ holds by $(\alpha+1_q)\cdot(\langle 0_m+1_{n+p}, 1_m+0_{n+p}\rangle+1_q)=\langle 0_m+1_n, 1_m+0_n\rangle+1_{p+q}$. Thus, $\langle f_1, h_1, g_1\rangle^\dagger\theta\,\theta\langle f_1, h_1, \bar{g}(1_{n+p}+0_m+1_q)\rangle^\dagger$ by the previous case. From this the result follows by

$$(B_6)\colon\ \langle f, g, h\rangle^\dagger=\alpha\alpha^{-1}\langle f, g, h\rangle^\dagger=$$

$$=\alpha(\alpha^{-1}\langle f, g, h\rangle(\alpha+1_q))^\dagger=\alpha\langle f_1, h_1, g_1\rangle^\dagger\theta\alpha\langle f_1, h_1, \bar{g}(1_{n+p}+0_m+1_q)\rangle^\dagger=$$

$$=(\alpha\langle f_1, h_1, \bar{g}(1_{n+p}+0_m+1_q)\rangle(\alpha^{-1}+1_q))^\dagger=\langle f, \bar{g}(1_n+0_m+1_{p+q}), h\rangle^\dagger.$$

Let $\tau\colon [r]\to[r]$ be any bijection. We shall denote by $\bar{\pi}_\tau\colon 1\to 2^r\in T_\Pi$ the balanced tree visualized in the following figure:



In the case that τ is the identity mapping, the index τ will be omitted in $\bar{\pi}_\tau$.

Lemma 3. For any $f\colon 1\to p\in\overline{T}_\Pi$ there exists a (unique) base element $\varrho\colon 2^r\to p$ with $f\theta\bar{\pi}\varrho$.

Proof. This statement is well-known. In spite of this, for the sake of completeness, a proof will be outlined here. We shall show a little bit more than it is stated by our lemma. Namely, we show that for any $f\colon 1\to p\in\overline{T}_\Pi$ and bijective $\tau\colon [r]\to[r]$ there is a base element $\varrho\colon 2^r\to p$ with $f\theta\bar{\pi}_\tau\varrho$.

If $f=x_i$ ($i\in[p]$) then put $\varrho=\langle\overbrace{\pi_p^i, \ldots, \pi_p^i}^{2^r\text{-times}}\rangle$. Then $f\theta\bar{\pi}_\tau\varrho$ follows by applications of (C_1). We proceed by structural induction of f. Suppose that $f=\pi_i f_1 f_2$, $f_1\theta\bar{\pi}_\alpha\varrho_1$

and $f_2\theta\bar{\pi}_\alpha\varrho_2$ where $i\in[r]$ and $\alpha\colon [r]\to[r]$ is any bijection with $\alpha(1)=i$. Then, by (C_3), we obtain $f\theta\bar{\pi}_\alpha\varrho'$ for a suitable $\varrho'\colon 2^r\to p$. However, $\bar{\pi}_\alpha\varrho'\theta\bar{\pi}_\tau\varrho$ holds by (C_2) for a satisfactory choise of ϱ.

Lemma 4. $(f+0_p)^\dagger\theta\perp_{np}$ holds for every $f\colon n\to n\in R_\Sigma$.

Proof. By (B_2) it is enough to deal with the case $p=0$. If $n=0$ then the statement is obviously valid by $R_\Sigma(0,p)=\{0_p\}$. Assuming $n>0$ we have $f=\langle f_1,f_2\rangle$ where $f_1=(1_1+0_{n-1})f$, $f_2=(0_1+1_{n-1})f$. Thus, by (B_3), $f^\dagger=\langle h^\dagger,(f_2\varrho)^\dagger h^\dagger\rangle$, where $\varrho=\langle 0_{n-1}+1_1,1_{n-1}+0_1\rangle$, $h=f_1\langle 1_1,(f_2\varrho)^\dagger\rangle$. As $h\in R_\Sigma(1,1)$, $h^\dagger\theta\perp$ holds by (C_4). Therefore, $\pi_n^1 f^\dagger\theta\perp$. From this the result follows by (B_6).

Lemma 5. Given $f\colon n\to n+p\in T_\Pi$ there exists a $g\colon n\to p\in T_\Pi$ with $f^\dagger\theta g$.

Proof. The statement is obvious if $n=0$. Now assume that $n>0$ and proceed by induction on n. Define $f_1=(1_1+0_{n-1})f$, $f_2=(0_{n-1}+1_1)f$. By (C_5) and the induction hypothesis there exist $\bar{f}_1\colon 1\to n-1+p$, $\bar{f}_2\colon n-1\to 1+p$ with $f_1^\dagger\theta\bar{f}_1$ and $(f_2(\langle 0_{n-1}+1_1,1_{n-1}+0_1\rangle+1_p))^\dagger\theta\bar{f}_2$. Therefore, $f^\dagger\theta\langle 0_1+\bar{f}_1,\bar{f}_2(1_1+0_{n-1}+1_p)\rangle^\dagger$ holds by Lemma 2. By identity (B_7),

$$\langle 0_1+\bar{f}_1,\bar{f}_2(1_1+0_{n-1}+1_p)\rangle^\dagger=\langle\bar{f}_1\langle\bar{f}_2,0_1+1_p\rangle(1_1+0_{n-1}+1_p),\bar{f}_2(1_1+0_{n-1}+1_p)\rangle^\dagger.$$

Now let $\bar{h}=\bar{f}_1\langle\bar{f}_2,0_1+1_p\rangle$ and apply identity (B_8): $(\bar{h}(1_1+0_{n-1}+1_p))^\dagger=0_{n-1}+\bar{h}^\dagger$. As $\bar{h}\in T_\Pi(1,1+p)$ there is an element $h\colon 1\to p\in T_\Pi$ with $\bar{h}^\dagger\theta h$. Thus,

$$f^\dagger\theta\langle 0_n+h,\bar{f}_2(1_1+0_{n-1}+1_p)\rangle^\dagger$$

is valid by Lemma 2. Put $g=\langle h,\bar{f}_2\langle h,1_p\rangle\rangle$. Then, by (B_1), (B_3) and (B_8), $\langle 0_n+h,\bar{f}_2(1_1+0_{n-1}+1_p)\rangle^\dagger=(0_n+g)^\dagger=g$. As $g\in T_\Pi(n,p)$, this proves Lemma 5.

Definition 1. Let $f=\langle f_1,\ldots,f_n\rangle\colon n\to n+p\in T_\Sigma$ and let $i,j\in[n]$ be arbitrary. We say that f_i directly depends on f_j if there is an occurrence of variable x_j in f_i, i.e., $f_i(w)=x_j$ holds for some $w\in N^*$. The dependency relation is the transitive closure of direct dependency. A component f_i is called coaccessible provided that either there is an occurrence of a variable from $\{x_{n+1},\ldots,x_{n+p}\}$ in f_i or there is an integer j with f_i depends on f_j and f_j is coaccessible.

Lemma 6. Suppose that $f\colon n\to n+p\in T_\Sigma$. Then there is an element $g\colon n\to n+p\in T_\Sigma$ which only contains coaccessible or undefined components, and such that $f^\dagger\theta g^\dagger$ holds.

Proof. Put $f=\langle f_1,\ldots,f_n\rangle$ and let $f_{i_1},\ldots,f_{i_m}$ $(1\leqq i_1<\ldots<i_m\leqq n)$ be all those components of f which are not coaccessible. First suppose that $i_j=j$ holds for each $j\in[m]$. In this case there is an element $a\colon m\to m\in T_\Sigma$ with $(1_m+0_{n-m})f=a+0_{n-m+p}$. Thus, $f^\dagger\theta g^\dagger$ holds by Lemma 4 and Lemma 2, where

$$g=\langle\perp_{mn+p},(0_m+1_{n-m})f\rangle.$$

On the other hand, g only contains coaccessible or undefined components.

The general case, where $i_1,\ldots,i_m$ are arbitrary, is reducible to the previous one by (B_6).

Definition 2. An element $a\colon n\to n+p\in T_\Sigma$ $(n\geqq 1)$ is in normal form provided that each of its components $\pi_n^i a$ has one of the following four forms for every $i\in[n]$:

(i) $\pi_n^i a=\bar{\pi}\varrho+0_p$ where $\varrho\colon 2^r\to m$ is base,
(ii) $\pi_n^i a=\omega\varrho+0_p$ where $\omega\in\Omega$ and $\varrho\colon 1\to n$ is base,
(iii) $\pi_n^i a=\perp_{1n+p}$,
(iv) $\pi_n^i a=0_n+\varrho$ where $\varrho\colon 1\to p$ is base.

Furthermore, a is required to satisfy all conditions (v), (vi), (vii) and (viii) as well:

(v) if $\pi_n^i a$ is of type (i) then $\pi_n^j a$ has to have one of the forms (ii), (iii) or (iv) for each $j\in\varrho([2^r])$,
(vi) if $\pi_n^i a$ is of type (ii) then $\pi_n^{\varrho(i)}a$ is of type (i),
(vii) $\pi_n^1 a$ is of type (i),
(viii) every component $\pi_n^i a$ of type (ii) is coaccessible.

Lemma 7. For every $f\colon 1\to p\in R_\Sigma$ there is an element $y\colon k\to k+p$ in normal form such that $f\theta\pi_k^1 y^\dagger$.

Proof. By a simple modification of Theorem 2.5.1 in [5] we obtain that there is an element $a\colon n\to n+p\in T_\Sigma$ with $f=\pi_n^1 a^\dagger$ and such that each of its components $\pi_n^i a$ $(i\in[n])$ has one of the three forms (ii), (iii) or (iv), or $\pi_n^i a=a_i+0_p$ holds for some $a_i\in\overline{T}_\Pi(1,n)$. Furthermore, by identity (B_9), we may assume a to satisfy the following modified version of (vi): if $\pi_n^i a=\omega\varrho+0_p$ for some $\omega\in\Omega$ and $\varrho\colon 1\to n$ then $\pi_n^{\varrho(1)}a=\pi_{n+p}^j$ is valid for an integer $j\in[n]$. Finally, we may assume that $\pi_n^n a=\perp_{1n+p}$ since otherwise a can be replaced by $\langle a(1_n+0_1+1_p),\perp_{1n+1+p}\rangle$ (cf. (B_{10})).

Let $i_1,\ldots,i_m\in[n]$ $(i_1<\ldots<i_m)$ be all those indices for which $\pi_n^{ij}a$ is in $\overline{T}_\Pi(1,n+p)-\{\pi_{n+p}^{n+1},\ldots,\pi_{n+p}^{n+p}\}$. First suppose that $i_j=j$ holds for each j. Put $b_i=(1_m+0_{n-m})a$, $c=(0_m+1_{n-m})a$. Then $a=\langle b_1,c\rangle$ holds obviously. Observe that $b_1=\bar{b}_1+0_p$ holds for some $\bar{b}_1\colon m\to n\in\overline{T}_\Pi$. Therefore, by Lemma 5 and (B_2), there exists $b_2\colon m\to n-m\in T_\Pi$ with $b_1^\dagger\theta b_2+0_p$. Thus, by Lemma 2, $\langle 0_m+b_2+0_p,c\rangle^\dagger\theta a$. There is an element $b_3\colon m\to n-m+1\in\overline{T}_\Pi$ with $b_2=b_3(1_{n-m}+\perp)$. Put $b_4=b_3\langle 1_{n-m},\pi_{n-m}^{n-m}\rangle$. Then $\langle 0_m+b_2+0_p,c\rangle^\dagger=\langle 0_m+b_4+0_p,c\rangle^\dagger$ follows by (B_7) and $\pi_n^n a=\perp_{1n+p}$. On the other hand, by Lemma 3, we have

$$\langle 0_m+b_4+0_p,c\rangle^\dagger\theta\langle 0_m+b_5+0_p,c\rangle^\dagger$$

for some $b_5\colon m\to n-m$ whose each component is of type $\bar{\pi}\varrho$ for a suitable base element $\varrho\colon 2^r\to n-m$. Next, by an application of Lemma 6, we get an element $d\colon n\to n+p$ whose each component is either coaccessible or undefined, and $\langle 0_m+b_5+0_p,c\rangle^\dagger\theta d^\dagger$ holds. It follows from the proof of Lemma 6 that d satisfies all conditions in Definition 2 except possibly (vii). If d does not satisfy (vii) then let $g=\langle\bar{\pi}\varrho+0_p,0_1+d\rangle$ where $\varrho\colon 2^r\to n+1$ is defined by $\varrho(i)=2$, $i\in[2^r]$. Otherwise put $g=d$. In both cases g is in normal form and $\pi_n^1 g^\dagger\theta f$ (cf. (B_{11}) and Lemma 3).

The general case, i.e. where $i_1,\ldots,i_m$ are arbitrary, is reducible to the special one above (cf. (B_6)).

Lemma 8. Let $a\colon n\to n+p\in T_\Sigma$ and $b\colon m\to m+p\in T_\Sigma$ be in normal form. Then $\pi_1^n a^\dagger\equiv\pi_m^1 b^\dagger$ if and only if $\pi_n^1 a^\dagger=\pi_m^1 b^\dagger$.

Proof. Sufficiency is obvious. Conversely, let $f=\pi_n^1 a^\dagger$, $g=\pi_m^1 b^\dagger$ and suppose that $f\equiv g$. Define $\bar{f}\colon N^*\multimap\Sigma^*$ by $\bar{f}(\lambda)=f(\lambda)$ and $\bar{f}(wi)=\bar{f}(w)f(i)$ if $w\in N^*$ and

$i\in[n]$. As $f\equiv g$ and a, b are in normal form, $f^{-1}(x_i)=g^{-1}(x_i)$, i.e. $\{w|f(w)=x_i\}=\{w|g(w)=x_i\}$ holds for any $i\in[p]$. Furthermore, if $w\in\bigcup(f^{-1}(x_i)|i\in[p])$ then $\bar{f}(w)=\bar{g}(w)$ where $\bar{g}$ is similarly defined with respect to g as $\bar{f}$ was defined with respect to f. The above equalities are essentially known from [4] (cf. also [11], [12]).

Suppose that $f\neq g$. Then, as $f^{-1}(x_i)=g^{-1}(x_i)$ holds for each $i\in[p]$, there is a string $w\in N^*$ with $f(w)\neq g(w)$ and both $f(w)$ and $g(w)$ are in Ω or one of them is undefined. Thus two cases arise. However, similar order of ideas yields a contradiction in both cases. Therefore we assume that $f(w)\in\Omega$. By the last condition in the definition of normal forms, there is a string $v\in N^*$ with $wv\in\bigcup(f^{-1}(x_i)|i\in[p])$. As $f(w)\neq g(w)$ also $\bar{f}(wv)\neq\bar{g}(wv)$. This is a contradiction.

Now we are ready to state our

Theorem. $\theta=\equiv$.

Proof. $\theta\subseteq\equiv$ is valid by Lemma 1. Conversely, it is enough to show that $f\equiv g$ implies $f\theta g$ for arbitrary $f, g\in R_\Sigma(1,p)$. But this is immediate by Lemma 7 and Lemma 8.

An equational characterization of the strong equivalence of Elgot's flow-chart schemata was given in [6]. Here we present an equational characterization for the weak equivalence. An extended abstract of this paper has been already appeared in [14].

References

[1] Bloom, S. L. and C. C. Elgot, The existence and construction of free iterative theories, J. Comput System Sci. v. 12, 1976, pp. 305—318.
[2] Bloom, S. L., C. C. Elgot and J. B. Wright, Vector iteration in pointed iterative theories, SIAM J. Comput., v. 9. 1980, pp. 525—540.
[3] Elgot, C. C., Monadic computation and iterative algebraic theories, in Logic Colloquium'73, eds. Rose, H. E. and J. C. Shepherdson, v. 80, Studies in Logic, North Holland, Amsterdam, 1975, pp. 175—230.
[4] Elgot, C. C., Structured programming with and without GO TO statements, IEEE Trans. Soft. Engineering, v. SE—2, 1976, pp. 41—53.
[5] Elgot, C. C., S. L. Bloom and R. Tindell, On the algebraic structure of rooted trees, J. Comput. System Sci., v. 16, 1978, pp. 362—399.
[6] Ésik, Z., Identities in iterative and rational algebraic theories, Computational Linguistics and Computer Languages, v. XIV, 1980, pp. 183—207.
[7] Ésik, Z., On generalized iterative theories, Computational Linguistics and Computer Languages, v. XV, 1982, pp. 95—110.
[8] Ésik, Z., Algebras of iteration theories, J. Comput. Syst. Sci. 27 (1983), 291—303.
[9] Goguen, J. A., J. W. Thatcher, J. W. Wagner and J. B. Wright, Initial algebraic semantics and continuous algebras, J. Assoc. Comput. Mach., v. 24., 1977, pp. 68—95.
[10] Lawvere, F. W., Functorial semantics of algebraic theories, Proc. Nat. Acad. Sci. USA 50, 1963, pp. 869—872.
[11] Luckham A., A. Park and A. Paterson, On formalized computer programs, J. Comput. Syst. Sci., v. 4., 1970, pp. 220—249.
[12] Rutledge, J. D., On Ianov's program schemata, J. Ass. Comput. Mach., v. 11., 1964, pp. 1—9.
[13] Wagner, E. G., J. B. Wright, J. A. Goguen and J. W. Thatcher, Some fundamentals of order-algebraic semantics, in Proceedings, Mathematical Foundations of Computer Science, 1976, ed. Mazurkievicz, A., LNCS 45, 1976, pp. 151—168.
[14] Ésik, Z., On Elgot's flow-chart schemes, Syst. Theoretical Aspects in Comp. Sci., Salgótarján, 1982, pp. 99—102.

Received Oct. 12, 1982

Atomic characterizations of uniform multi-pass attribute grammars

By E. GOMBÁS and M. BARTHA

1. Introduction

Several reasonable classes of attribute grammars can be defined based on the concept of computation sequence [1]. A computation sequence for a derivation tree is intended to describe a systematic evaluation of all the attributes of the tree without violating their dependencies. The attributes are evaluated during a walk through the tree. This walk starts at the root, and once it arrives at (enters) a node, it must return to that node later for exiting it. Between any successive entering and exiting a node — this period is called a visit to the node — the walk can make several visits to the sons of the node. If in any derivation tree, the number of visits to a node required to evaluate all the attributes of it, and the set of attributes of the node evaluated in each of these visits can both be determined in a top-down manner, i.e. independently of the subtree below that node, then the grammar is called uniform [5]. It was proved in [5] that an attribute grammar (*AG*) is uniform iff it is absolutely noncircular (anc). The latter property is investigated e.g. in [3], where an efficient evaluator is given for these grammars. As the anc property can be decided in polynomial time, the class of uniform *AG* is practically more interesting than the class of simple multi-visit *AG* introduced in [1]. (Recall from [1] that the problem deciding whether an *AG* is simple multi-visit is *NP*-complete.) However, if we ask whether an *AG* is uniform m-visit for a fixed $m \in \mathbf{N}$, then the answer cannot be given in polynomial time, generally. Thus, to answer this question in polynomial time we have to restrict ourselves to a smaller class of *AG*. In this paper we investigate the class of uniform multi-pass *AG*. A pass to a node is a visit such that during it each son of the node is visited exactly once in a left-to-right order. We present a natural characterization of this class and give an algorithm that provides the minimal number m for which an *AG* is uniform m-pass in polynomial time.

2. Definitions and basic concepts

An attribute grammar [4] $\mathcal{G}$ consists of the following objects.

(i) A reduced context-free grammar $G=(T, N, P, Z)$.

(ii) A finite nonempty set A such that $A=A_s \cup A_i$ and $A_s \cap A_i=\emptyset$. The elements of A_s and A_i are called synthesized (s-) and inherited (i-) attributes, respectively.

(iii) A function v which assigns each nonterminal $F\in N$ a nonvoid subset of A. We assume that the start symbol Z has only s-attributes and it does not occur on the right-hand side of any production. $S(F)$ and $I(F)$ will denote $v(F)\cap A_s$ and $v(F)\cap A_i$, respectively, and an occurrence of an attribute $a\in v(F)$ will often be referenced as $a(F)$.

(iv) A set $V(a)$ of possible values for each attribute a.

(v) A set r_p of semantic rules associated with each production $p\in P$. If p: $F_0 \to w_0 F_1 \dots F_k w_k$ ($F_i\in N$, $w_i\in T^*$), then a rule of r_p is a formal equation:

$$a_0(F_{i_0}) = f\big(a_1(F_{i_1}), \dots, a_m(F_{i_m})\big),$$

where $0\leqq i_j\leqq k$ ($0\leqq j\leqq m$), $a_j\in v(F_{i_j})$ and f: $V(a_1)\times\dots\times V(a_m)\to V(a_0)$ is a (computable) function. This equation is interpreted by saying that $a_0(F_{i_0})$ depends on $a_1(F_{i_1}), \dots, a_m(F_{i_m})$ in p by f. We assume that $\mathcal{G}$ is in Bochmann normal form, i.e. r_p defines all and only the occurrences of attributes $S(F_0)\cup\big(\cup(I(F_j)|\, j\in[k])\big)$ using as arguments only of the occurrences of $I(F_0)\cup\big(\cup(S(F_j)|\, j\in[k])\big)$. ($[k]$ denotes the set $\{1, 2, \dots, k\}$.)

D_F will denote the set of derivation trees with root labelled by F. Trees of D_Z are called complete derivation trees. If t is a tree representing a derivation $F\overset{*}{\underset{G}{\Rightarrow}}\alpha$, where α is not necessarily a terminal string, then t is called a cut; in notation, $t\in C_F$. Clearly we have $D_F\subseteqq C_F$ for all $F\in N$. By a node of t we always mean a nonterminal node, and if there is no danger of confusion, we identify the node with its label. U_t, $U_t(F)$ and $rt(t)$ will denote the set of all nodes of t, the set of all F-labelled nodes of t and the root of t, respectively.

The semantic rules are used to assign meanings to derivation trees of G in the following way. Let t be a complete derivation tree, $u\in U_t$, and assume that p: $F_0 \to w_0 F_1 \dots F_k w_k$ is the production applied at u. For each $a_0\in S(F_0)$, the function f occurring in the rule $a_0(F_0)=f\big(a_1(F_{i_1}), \dots, a_m(F_{i_m})\big)$ can be used to determine the value of a_0 at u when the values of all the neighbouring attributes $a_1(F_{i_1}), \dots, a_m(F_{i_m})$ have been determined. Similarly, the rule with left-hand side $b(F_j)$ $(j\in[k], b\in I(F_j))$ can be used to determine the value of attribute b at the j-th son of u. If it is possible to determine the values of all the attributes at any node of t in the above way, then the meaning of t is the set $\big\{(u, \{v_a(u)|a\in v(u)\})|u\in U_t$ and $v_a(u)$ is the value of attribute a at $u\big\}$.

If all the complete derivation trees have a meaning, then $\mathcal{G}$ is called well-defined or noncircular.

The dependency graph for the production p: $F_0 \to w_0 F_1 \dots F_k w_k$ (denoted by $dp(p)$) has as nodes the disjoint union of $v(F_i)$ $0\leqq i\leqq k$, and there is an arc from $a_1(F_{i_1})$ to $a_2(F_{i_2})$ iff $a_2(F_{i_2})$ depends on $a_1(F_{i_1})$ in p. A graph with nodes $v(F)$ $(F\in N)$ and some arcs is called a dependency graph (d-graph) for F. For p: $F_0 \to w_0 F_1 \dots F_k w_k$ and d-graphs $\gamma_1, \dots, \gamma_k$ for $F_1, \dots, F_k$ define the substitu-

tion $dp(p)(\gamma_1, ..., \gamma_k)$ of $\gamma_1, ..., \gamma_k$ into $dp(p)$ by adding all the arcs of γ_i $i \in [k]$ to $dp(p)$, i.e. fitting γ_i on $dp(p)|v(F_i)$. This substitution induces a d-graph for F_0 by restricting the transitive closure of $dp(p)(\gamma_1, ..., \gamma_k)$ to $v(F_0)$. Now, the induced dependency graph for symbol $F \in N$ $(ids(F))$ is defined as the least d-graph for F such that for any production p: $F_0 \to w_0 F_1 \dots F_k w_k$, the d-graph for F_0 induced by the substitution $dp(p)(ids(F_1), ..., ids(F_k))$ is a subgraph of $ids(F_0)$. The induced dependency graph for production p is $idp(p) = dp(p)(ids(F_1), ..., ids(F_k))$. We write $a_1(F_{i_1}) <_p a_2(F_{i_2})$ if there is a nonempty path in $idp(p)$ from $a_2(F_{i_2})$ to $a_1(F_{i_1})$. Similarly, $a <_F b$ denotes that there is an arc in $ids(F)$ from b to a.

The induced dependency graph for a cut t $(idt(t))$ is obtained by pasting together the idp's of all the productions t consists of. Let u_1 and u_2 be two nodes of t, $a_1 \in v(u_1)$, $a_2 \in v(u_2)$. As above, $a_1(u_1) <_t a_2(u_2)$ denotes that there is a path in $idt(t)$ from $a_2(u_2)$ to $a_1(u_1)$. We write $a_1(u_1) <_t^R a_2(u_2)$ if this path contains an R-arc. Recall from [2] that an R-arc leads to an i-attribute of a node from an s-attribute of itself or one of its right neighbours. G is called absolutely noncircular (anc) if $<_t$ is a strict partial order for every cut t.

The following AG will be used as an example throughout the paper. G has five nonterminals with attributes:

$$v(Z) = \{a_0\}, \quad v(A) = \{a_0, a_1, a_2, b_1, b_2\}$$
$$v(B) = \{a_1, a_2, b_1, b_2\}, \quad v(C) = \{a_1, b_1\}, \quad v(D) = \{a_1, a_2, b_1, b_2\}.$$

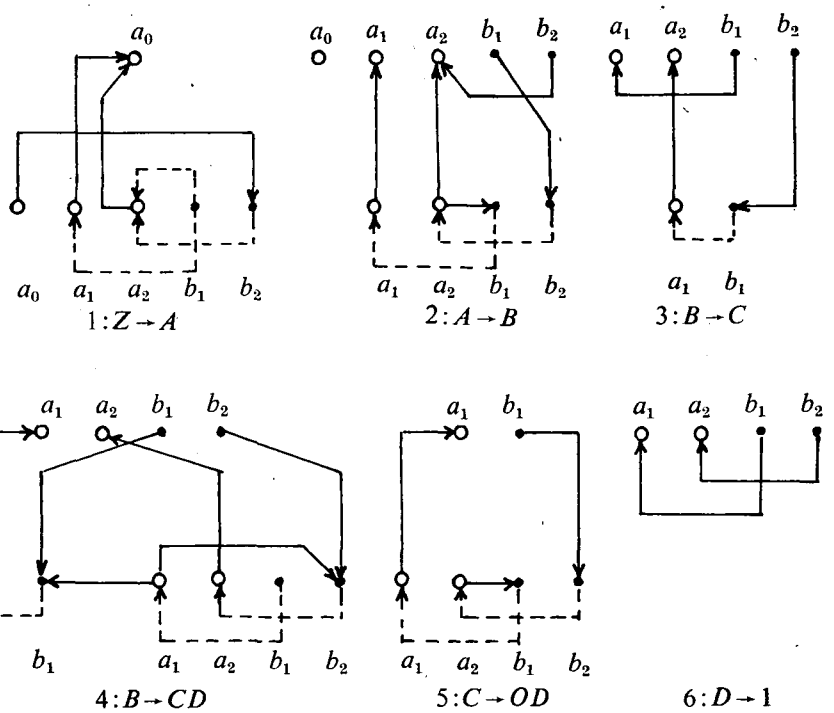


Fig. 1.

$A_s=\{a_0, a_1, a_2\}$, $A_i=\{b_1, b_2\}$. The productions with the corresponding *dp*-graphs are listed in Fig. 1. Dotted lines denote *idp*-arcs, 0 and 1 are terminal symbols.

A partial computation sequence for a derivation tree $t\in D_F$ is a sequence h of so called basic actions ([1]), where each basic action is either the evaluation of some *i*-attributes of a node, called entering the node, or the evaluation of some *s*-attributes of a node, called exiting the node. Thus, a basic action can be represented by a basic action symbol (ba-symbol) $i(u, B)$ or $s(u, A)$, where $u\in U_t$, $A\subseteq S(u)$ and $B\subseteq I(u)$. The order of evaluation is systematic and it cannot violate the dependencies of the attributes. By this we mean that h must obey the following restrictions.

1. The first and the last ba-symbol of h is $i(rt(t), B)$ and $s(rt(t), A)$, respectively, where $A\subseteq S(F)$ and $B\subseteq I(F)$.

2. For any two contiguous ba-symbols $\ldots x_1(u_1, A_1)x_2(u_2, A_2)\ldots$ in h, one of the following conditions holds.

(i) u_2 is a son of u_1 and $x_1=x_2=i$,
(ii) u_2 is the father of u_1 and $x_1=x_2=s$,
(iii) u_2 is a brother of u_1 and $x_1=s$, $x_2=i$,
(iv) $u_2=u_1$ and $x_1\neq x_2$.

3. For every $u\in U_t$, if

$$i(u, B_1)s(u, A_1)\ldots i(u, B_m)s(u, A_m)$$

is the sequence of all the ba-symbols for u occurring in h (from left to right), then $(B_1\cup A_1, \ldots, B_m\cup A_m)$ is an ordered subpartition of $v(u)$. This subpartition will be denoted by $E_h(u)=(E_1(u), \ldots, E_m(u))_h$ or $E(u)$ if h is understood. By an ordered subpartition of a set C we mean a sequence of sets $(C_1, \ldots, C_m)$ such that

$$\bigcup_{i=1}^{m} C_i \subseteq C \quad \text{and} \quad C_i\cap C_j=\emptyset \quad \text{if} \quad 1\leqq i\neq j\leqq m.$$

4. For any production p, consider an arbitrary occurrence of p in t, and let u_1, u_2 and a_1, a_2 such nodes and attributes of this occurrence that $a_2(u_2)$ depends on $a_1(u_1)$ in p. If $a_2(u_2)$ occurs in h (i.e. there exists a ba-symbol $x(u_2, A)$ in h with $a_2\in A$), then so does $a_1(u_1)$, and the occurrence of $a_1(u_1)$ precedes that of $a_2(u_2)$.

If $E_h(u)$ is a complete partition (i.e. $\cup E_h(u)=v(u)$) for all $u\in U_t$, then h is a (total) computation sequence. If h satisfies 1, 2 and 3 only, then it is called a walk. A walk h is a pass if:

— each node is entered and exited (i.e. visited) exactly once;
— during the visit to a node, all its sons are visited in a left to right order.

h is an m-pass walk ($m\in\mathbf{N}$) if $h=h_1\ldots h_m$ and h_i is a pass for all $i\in[m]$.

A ba-symbol $x(u, A)$ is empty if $A=\emptyset$. A pass is called empty if all the ba-symbols occurring in it are empty.

Example 2.1. Let t be the complete derivation tree of our example AG illustrated on the left-hand side of Fig. 2. The graph on the right-hand side indicates the dependencies between the attributes of t. A 4-pass computation sequence for t is the following.

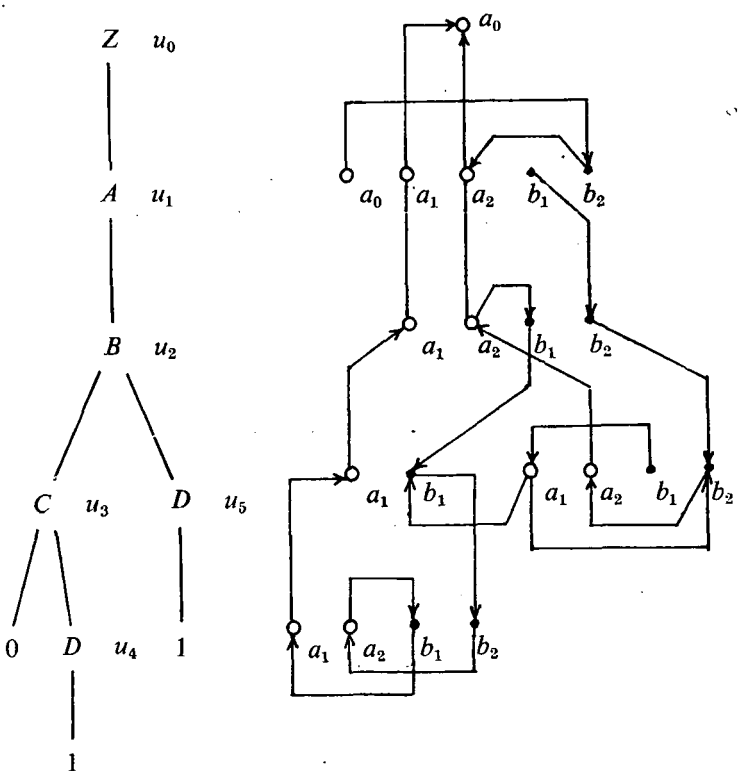


Fig. 2.

$h=h_1h_2h_3h_4$, where

$$h_1 = i(u_0,\emptyset)\,i(u_1,\{b_1\})\,i(u_2,\{b_2\})\,i(u_3,\emptyset)\,i(u_4,\emptyset)\,s(u_4,\emptyset)\,s(u_3,\emptyset)$$
$$i(u_5,\{b_1\})\,s(u_5,\{a_1\})\,s(u_2,\emptyset)\,s(u_1,\{a_0\})\,s(u_0,\emptyset),$$
$$h_2 = i(u_0,\emptyset)\,i(u_1,\emptyset)\,i(u_2,\emptyset)\,i(u_3,\emptyset)\,i(u_4,\emptyset)\,s(u_4,\emptyset)\,s(u_3,\emptyset)$$
$$i(u_5,\{b_2\})\,s(u_5,\{a_2\})\,s(u_2,\{a_2\})\,s(u_1,\emptyset)\,s(u_0,\emptyset),$$
$$h_3 = i(u_0,\emptyset)\,i(u_1,\emptyset)\,i(u_2,\{b_1\})\,i(u_3,\{b_1\})\,i(u_4,\{b_2\})\,s(u_4,\{a_2\})$$
$$s(u_3,\emptyset)\,i(u_5,\emptyset)\,s(u_5,\emptyset)\,s(u_2,\emptyset)\,s(u_1,\emptyset)\,s(u_0,\emptyset),$$
$$h_4 = i(u_0,\emptyset)\,i(u_1,\{b_2\})\,i(u_2,\emptyset)\,i(u_3,\emptyset)\,i(u_4,\{b_1\})\,s(u_4,\{a_1\})$$
$$s(u_3,\{a_1\})\,i(u_5,\emptyset)\,s(u_5,\emptyset)\,s(u_2,\{a_1\})\,s(u_1,\{a_1,a_2\})\,s(u_0,\{a_0\}).$$

Let p: $F_0 \to w_0F_1 \dots F_kw_k \in P$ and $t_j \in D_{F_j}$ for each $j \in [k]$. Let

$$\pi = (A_1 \cup B_1, \dots, A_m \cup B_m)$$

be an ordered subpartition of $v(F_0)$ with $A_i \subseteqq S(F_0)$, $B_i \subseteqq I(F_0)$ $(i \in [m])$, and let $h_j = h_1^{(j)} \dots h_{m_j}^{(j)}$ be an m_j-pass walk $(m_j \leqq m)$ for each t_j. $\pi \circ (h_1, \dots, h_k)$ will denote

the m-pass walk $i(u_0, B_1)h_1^{(1)}\dots h_1^{(k)}s(u_0, A_1)\dots i(u_0, B_m)h_m^{(1)}\dots h_m^{(k)}s(u_0, A_m)$ for $F_0(w_0t_1\dots t_kw_k)\in D_{F_0}$, where u_0 is the root and $h_i^{(j)}$ is empty if $i>m_j$.

The proof of the following two easy lemmas are left to the reader.

Lemma 2.1. Let t be a complete derivation tree and $u\in U_t(F)$. Denote $t'\in D_F$ the subtree of t below u, and let $h=h_1\dots h_m$ and $l'=l'_1\dots l'_m$ be m-pass computation sequences for t and t', respectively. Each h_i ($i\in[m]$) can be written in the form $h_i=\alpha_i l_i\beta_i$, where $l=l_1\dots l_m$ is an m-pass computation sequence for t'. If $E_h(u)=E_{l'}(rt(t'))$, then $h'=h'_1\dots h'_m$ — where $h'_i=\alpha_i l'_i\beta_i$ — is also an m-pass computation sequence for t.

Let $\pi_1=(A_1, \dots, A_n)$ and $\pi_2=(B_1, \dots, B_m)$ be two ordered subpartitions of a set D. Construct the ordered subpartition

$$\text{merge}\,(\pi_1, \pi_2)=(C_1, \dots, C_{\max(n,m)})$$

as follows:

(i) $C_1 = A_1\cup B_1$,

(ii) $C_{i+1} = A_{i+1}\cup B_{i+1}\setminus\bigcup_{j=1}^{i} C_j$ for each $1\leqq i < \max(n, m)$.

Lemma 2.2. Let h_i ($i=1, 2,$) be m_i-pass partial computation sequences for $t\in D_F$, $m=\max(m_1, m_2)$. Construct an m-pass walk merge (h_1, h_2) as follows. For each $u\in U_t$ let

$$E_{\text{merge}(h_1,h_2)}(u)=\text{merge}\,(E_{h_1}(u), E_{h_2}(u)).$$

Then merge (h_1, h_2) is a partial computation sequence.

Note that an m-pass walk for t is completely determined by the set $\{E(u)|u\in U_t\}$. The operation merge can be extended to any number of walks by:

$$\text{merge}\,(h_1, \dots, h_{n+1}) = \text{merge}\,(\text{merge}\,(h_1, \dots, h_n), h_{n+1}).$$

3. The atomic characterization and decidability results

An atomic pass-description for $\mathcal{G}$ is a five-tuple $\mathcal{D}=(\mathcal{A}, \mu, \chi, \varrho, \pi_0)$, where

(i) $\mathcal{A}$ is a finite nonempty set, called the set of atoms.

(ii) μ: $N\to P(\mathcal{A})$ assigns each nonterminal a subset of $\mathcal{A}$.

(iii) $\chi=\{\chi_F|F\in N\}$, where χ_F: $\mu(F)\to\mathcal{P}(v(F))$ is a function such that

$$\bigcup_{c\in\mu(F)}\chi_F(c) = v(F) \quad\text{and}\quad \chi_F(c_1)\cap\chi_F(c_2)=\emptyset \quad\text{if}\quad c_1\neq c_2.$$

(iv) $\varrho=\{\varrho_p|p\in P\}$ is a family of mappings such that if p: $F_0\to w_0F_1\dots F_kw_k$, then

$$\varrho_p\colon \mu(F_0)\to\mathcal{P}\Big(\bigcup_{j=1}^{k}(\{j\}\times\mu(F_j))\Big).$$

As in the case of attributes, $c(F)$ indicates an occurrence of an atom $c\in\mu(F)$, and we prefer the notation $c(F_j)\in\varrho_p(c_0)$ to $(j, c)\in\varrho_p(c_0)$ if no confusion arises.

(v) π_0 is an ordered partition of $\mu(Z)$.

Let π be an ordered subpartition of $\mu(F)$. We say that an m-pass partial computation sequence h for $t\in D_F$ respects $\mathcal{D}/\pi$ if it satisfies the following conditions.

1. For every node $u \in U_t(Y)$ $(Y \in N)$ there exists an ordered subpartition $\pi_h(u) = (\mathscr{A}_1, \ldots, \mathscr{A}_m)$ of $\mu(Y)$ such that

$$(E_i(u))_h = \bigcup(\chi_Y(c) | c \in \mathscr{A}_i) \quad \text{for all} \quad i \in [m].$$

2. For any production p: $F_0 \to w_0 F_1 \ldots F_k w_k$ consider an arbitrary occurrence of p in t. Let $u_0, u_1, \ldots, u_k$ be the corresponding nodes of t, respectively. If $\pi_h(u_l) = (\mathscr{A}_1^{(l)}, \ldots, \mathscr{A}_m^{(l)})$ $(0 \leqq l \leqq k)$, then

a) for every $i \in [m]$ and $j \in [k]$

$$\{c(F_j) | c \in \mathscr{A}_i^{(j)}\} \subseteq \varrho_p(\mathscr{A}_i^{(0)}) \cap \mu(F_j);$$

b) if $c(F_j) \in \varrho_p(c_1) \cap \varrho_p(c_2)$ for some $c_1 \neq c_2$ such that $c \in \mathscr{A}_i^{(j)}$ and $c_l \in \mathscr{A}_{i_l}^{(0)}$ $(l = 1, 2)$, then $i \leqq \min(i_1, i_2)$.

3. $\pi_h(rt(t)) = \pi$.

Definition 3.1. $\mathscr{G}$ is atomic m-pass with respect to $\mathscr{D}$ if every complete derivation tree has an m-pass computation sequence respecting $\mathscr{D}/\pi_0$. $\mathscr{G}$ is atomic multi-pass (amp) if there exist m and $\mathscr{D}$ such that $\mathscr{G}$ is atomic m-pass with respect to $\mathscr{D}$.

Example 3.1. In our example AG let $\mathscr{D} = (\mathscr{A}, \mu, \chi, \varrho, \pi_0)$, where

$$\mathscr{A} = \{d_1, d_2\} \cup \{(c_1, i) | i \in [3]\} \cup \{(c_2, i) | i \in [2]\} \cup$$
$$\cup \{(c_0, i),\ (c, i) | i \in [4]\};$$

$$\mu(Z) = \{(c_0, i) | i \in [4]\},\ \mu(A) = \{(c_0, 1),\ d_1\} \cup \{(c, i) | i \in [4]\},$$
$$\mu(B) = \{(c_1, i) | i \in [3]\} \cup \{(c_2, i) | i \in [2]\} \cup \{d_1, d_2\},$$
$$\mu(C) = \{(c_1, i) | i \in [2]\} \cup \{d_1\},\ \mu(D) = \{(c_1, 1),\ (c_2, 1)\};$$

$$\chi_Z(c_0, i) = \begin{cases} \emptyset & \text{if } i \leqq 3 \\ \{a_0\} & \text{if } i = 4 \end{cases} \quad \chi_A(c, i) = \begin{cases} \emptyset & \text{if } i \leqq 3 \\ \{a_1, a_2, b_2\} & \text{if } i = 4 \end{cases}$$

$$\chi_A(c_0, 1) = \{a_0\} \quad \chi_A(d_1) = \chi_B(d_1) = \{b_1\} \quad \chi_B(d_2) = \{b_2\}$$

$$\chi_B(c_1, i) = \begin{cases} \emptyset & \text{if } i \leqq 2 \\ \{a_1\} & \text{if } i = 3 \end{cases} \quad \chi_B(c_2, i) = \begin{cases} \emptyset & \text{if } i = 1 \\ \{a_2\} & \text{if } i = 2 \end{cases}$$

$$\chi_C(c_1, i) = \begin{cases} \emptyset & \text{if } i = 1 \\ \{a_1\} & \text{if } i = 2 \end{cases} \quad \chi_C(d_1) = \{b_1\}$$

$$\chi_D(c_1, 1) = \{a_1, b_1\} \quad \chi_D(c_2, 1) = \{a_2, b_2\};$$

$$\varrho_1(c_0, 1) = \{(c_0, 1)(A),\ d_1(A),\ (c, 1)(A)\},\quad \varrho_1(c_0, 2) = \{(c, 2)(A)\},$$
$$\varrho_1(c_0, 3) = \{(c, 3)(A)\},\quad \varrho_1(c_0, 4) = \{(c, 4)(A)\},$$
$$\varrho_2(c_0, 1) = \varrho_2(d_1) = \emptyset,\quad \varrho_2(c, 1) = \{d_2(B),\ (c_2, 1)(B)\},$$
$$\varrho_2(c, 2) = \{(c_2, 2)(B),\ (c_1, 1)(B)\},$$
$$\varrho_2(c, 3) = \{(c_1, 2)(B),\ d_1(B)\},\quad \varrho_2(c, 4) = \{(c_1, 3)(B)\},$$

$$\varrho_3(c_1, 1) = \varrho_3(c_1, 2) = \varrho_3(c_1, 3) = \emptyset,$$
$$\varrho_3(c_2, 1) = \{(c_1, 1)(C)\}, \quad \varrho_3(c_2, 2) = \{(c_1, 2)(C)\},$$
$$\varrho_4(d_1) = \varrho_4(d_2) = \emptyset, \quad \varrho_4(c_1, 1) = \{(c_1, 1)(D)\},$$
$$\varrho_4(c_1, 2) = \{d_1(C), (c_1, 1)(C)\}, \quad \varrho_4(c_1, 3) = \{(c_1, 2)(C)\},$$
$$\varrho_4(c_2, 1) = \{(c_1, 1)(D)\}, \quad \varrho_4(c_2, 2) = \{(c_2, 1)(D)\},$$
$$\varrho_5(d_1) = \emptyset, \quad \varrho_5(c_1, 1) = \{(c_2, 1)(D)\}, \quad \varrho_5(c_1, 2) = \{(c_1, 1)(D)\},$$
$$\varrho_6(c_1, 1) = \varrho_6(c_2, 1) = \emptyset;$$
$$\pi_0 = (\{(c_0, 1)\}, \{(c_0, 2)\}, \{(c_0, 3)\}, \{(c_0, 4)\}).$$

It is not difficult to check that the example AG is atomic 4-pass with respect to $\mathscr{D}$. For example, the computation sequence h of Example 2.1 respects $\mathscr{D}/\pi_0$.

Readers who are less familiar with attribute evaluation procedures are advised to read section 4 before going further.

Lemma 3.1. Let h_i $(i=1, 2, \ldots, n)$ be m_i-pass partial computation sequences for $t \in D_F$ respecting $\mathscr{D}/\pi_i$. Then merge $(h_i | i \in [n])$ respects $\mathscr{D}/\text{merge}\,(\pi_i | i \in [n])$.

Proof. Obvious.

Definition 3.2. $\mathscr{G}$ is absolutely non-R-recursive (anr) if it is anc and there is no cut t for which the following holds. There is a leaf $u \in U_t$ having the same label F as $u_0 = rt(t)$ and an attribute $a \in v(F)$ such that: $a(u_0) <_t^R a(u)$ or $a(u) <_t^R a(u_0)$.

Let $\mathscr{G}$ be anr. We shall prove that it is amp, too, moreover, we give an algorithm that provides the minimal number m for which $\mathscr{G}$ is atomic m-pass. In the first step the algorithm computes a relative difference $r(F, a, b)$ for each triple $F \in N$, $a <_F b$. $\{r(F, a, b) | F \in N, a <_F b\}$ is the system of minimal numbers such that for any p: $F_0 \to w_0 F_1 \ldots F_k w_k$ and idp (p)-path

$$a_0(F_0) <_p a_1(F_{i_1}) <_{F_{i_1}} b_1(F_{i_1}) <_p \ldots <_p a_n(F_{i_n}) <_{F_{i_n}} b_n(F_{i_n}) <_p b_0(F_0)$$

containing l R-arcs,

$$r(F_0, a_0, b_0) \geqq l + \sum_{j=1}^{n} r(F_{i_j}, a_j, b_j).$$

$r(F, a, b)$ expresses that, during an amp evaluation of derivation tree, at any F-labelled node, if $b(F)$ can be evaluated in the i-th pass, then $a(F)$ cannot be evaluated sooner than in the $i + r(F, a, b)$-th pass. This statement will be proved in Lemma 3.6. In the second step a pass-number $q(F, a)$ is computed for each $F \in N$, $a \in S(F)$ such that if $t \in D_F$ is an arbitrary derivation tree, and the values of all the i-attributes at the root are available, then — supposing an amp evaluation — $a(rt(t))$ can be evaluated in the $q(F, a)$-th pass, but not sooner in general. If all these numbers are finite, then the required atomic description can be constructed easily.

Algorithm 3.1. *Input:* An anr AG $\mathscr{G}$.
Output: pass-numbers $q(F, a)$ for each $F \in N$, $a \in S(F)$;
inherited pass-numbers $q_p(F_j, a, a_0)$ for each
p: $F_0 \to w_0 F_1 \ldots F_k w_k$, $j \in [k]$, $a \in v(F_j)$, $a_0 \in S(F_0)$ such that $a_0(F_0) <_p a(F_j)$.

(1) Compute for each $F\in N$, $a\prec_F b$ the number $r(F, a, b)$ as follows.
(a) Let $r_0(F, a, b)=0$ and set $i=0$.
(b) For each p: $F_0\to w_0 F_1 \dots F_k w_k\in P$
begin for each $b_0\in I(F_0)$
begin let $H_0=\{b_0(F_0)\}$, $x(b_0(F_0))=0$ and set $n=0$.
(i) Let

$$H_{n+1}=H_n\cup\{a(F_j)|a\in S(F_j),\ j\in[k],\ a(F_j)\prec_p b_0(F_0) \text{ and for any } a'(F_m)\in \cup$$
$$\cup(S(F_l)|l\in[k])\cup H_0,\ a(F_j)\prec_p a'(F_m)\prec_p b_0(F_0) \text{ implies that } a'(F_m)\in H_n\}.$$

If $H_{n+1}=H_n$, then goto step (ii), else for each $a(F_j)\in H_{n+1}\setminus H_n$ let

$$x(a(F_j))=\max(\max(r_i(F_j, a, b)+x(a'(F_m))|a(F_j)\prec_{F_j} b(F_j)\prec_p a'(F_m)\in H_n),$$
$$\max(r_i(F_j, a, b)+x(a'(F_m))+1|a(F_j)\prec_{F_j} b(F_j)\prec_p^R a'(F_m)\in H_n))$$

(note that $\max(\emptyset)=0$ by definition);
set $n=n+1$ and repeat step (i).
(ii) For each $a_0\prec_{F_0} b_0$ let

$$r_{i+1}^{(p)}(F_0, a_0, b_0)=\max(x(a(F_j))|a(F_j)\in H_n,\ a_0(F_0)\prec_p a(F_j));$$

end
end;

For each $F\in N$, $a\prec_F b$ let

$$r_{i+1}(F, a, b)=\max(r_{i+1}^{(p)}(F, a, b)|p\in P).$$

If $r_{i+1}(F, a, b)=r_i(F, a, b)$ for all $F\in N$, $a\prec_F b$, then let $r(F, a, b)=r_i(F, a, b)$, else set $i=i+1$ and repeat step (b).
(2) Compute for each $F\in N$, $a\in S(F)$ the number $q(F, a)$ as follows.
(c) Let $q_0(F, a)=1$ and set $i=0$.
(d) For each p: $F_0\to w_0 F_1 \dots F_k w_k$
begin let $M_0=\emptyset$ and set $n=0$.
(iii) Let

$$M_{n+1}=M_n\cup\{a(F_j)|\ j\in[k],\ a\in S(F_j) \text{ and for any } a'(F_m)\ (m\in[k],\ a'\in S(F_m)),$$
$$a(F_j)\prec_p a'(F_m) \text{ implies that } a'(F_m)\in M_n\}.$$

If $M_{n+1}=M_n$, then goto step (iv), else for each $a(F_j)\in M_{n+1}\setminus M_n$ let

$$y(a(F_j))=\max(q_i(F_j, a),\ \max(\max(r(F_j, a, b)+y(a'(F_m))|$$
$$a(F_j)\prec_{F_j} b(F_j)\prec_p a'(F_m)\in M_n),$$
$$\max(r(F_j, a, b)+y(a'(F_m))+1|a(F_j)\prec_{F_j} b(F_j)\prec_p^R a'(F_m)\in M_n)));$$

set $n=n+1$ and repeat step (iii).
(iv) For each $a_0=S(F_0)$ let

$$q_{i+1}^{(p)}(F_0, a_0)=\max(y(a(F_j))|a(F_j)\in M_n,\ a_0(F_0)\prec_p a(F_j))$$

end;

For each $F\in N$, $a\in S(F)$ let

$$q_{i+1}(F,a)=\max\left(q_{i+1}^{(p)}(F,a)\,|\,p\in P\right).$$

If $q_{i+1}(F,a)=q_i(F,a)$ for all $F\in N$, $a\in S(F)$, then let $q(F,a)=q_i(F,a)$, else set $i=i+1$ and repeat step (d).

(3) Compute for each $p\colon F_0\to w_0F_1\dots F_kw_k$, $j\in[k]$; $a\in v(F_j)$, $a_0\in S(F_0)$ such that $a_0(F_0)<_p a(F_j)$ the number $q_p(F_j,a,a_0)$ as follows.

Let $N_0=\{a_0(F_0)\}$, $q_p(F_0,a_0,a_0)=q(F_0,a_0)$ and set $n=0$.

(v) Let

$$N_{n+1}=N_n\cup\{a(F_j)\,|\,j\in[k],\ a\in v(F_j),\ a_0(F_0)<_p a(F_j)\ \text{and for any}\ a'(F_m)$$
$$(0\leqq m\leqq k,\ a'\in v(F_m)),\ a(F_0)\leqq_p a'(F_m)<_p a(F_j)\ \text{implies that}\ a'(F_m)\in N_n\}.$$

If $N_{n+1}=N_n$ then halt, else for each $a(F_j)\in N_{n+1}\setminus N_n$;

if $a\in I(F_j)$, then let

$$q_p(F_j,a,a_0)=\min\left(q_p(F_j,a',a_0)-r(F_j,a',a)\,|\,a'\in S(F_j),\ a'(F_j)\in N_n\right);$$

else (i.e. if $a\in S(F_j)$) let

$$q_p(F_j,a,a_0)=\min\big(\min\left(q_p(F_m,a',a_0)\,|\,a'(F_m)\in N_n,\ a'(F_m)<_p a(F_j)\right),$$
$$\min\left(q_p(F_m,a',a_0)-1\,|\,a'(F_m)\in N_n,\ a'(F_m)<_p^R a(F_j)\right)\big);$$

set $n=n+1$ and repeat step (v).

Example 3.2. Executing the algorithm on our example AG we get that:

$r(D,a_1,b_1)=r(D,a_2,b_2)=0$, $r(C,a_1,b_1)=1$,
$r(B,a_1,b_1)=r(B,a_2,b_2)=1$, $r(A,a_2,b_2)=0$,
$r(A,a_2,b_1)=1$, $r(A,a_1,b_1)=3$;
$q(D,a_1)=q(D,a_2)=1$, $q(C,a_1)=q(B,a_2)=2$,
$q(B,a_1)=3$, $q(A,a_2)=2$ (but $q_1(A,a_2,a_0)=4$),
$q(A,a_1)=4$, $q(A,a_0)=1$, $q(Z,a_0)=4$.

Lemma 3.2. For every $i\geqq 1$, $F\in N$ and $a<_F b$, if $r_i(F,a,b)>0$, then there exists $t\in D_F$ such that $a(rt(t))<_i^R b(rt(t))$.

Proof. Trivial, by construction.

Lemma 3.3. If $\mathcal{G}$ is anr, then the numbers $r(F,a,b)$ and $q(F,a)$ computed in steps (1) and (2) are all finite.

Proof. (a): $r(F,a,b)$ is finite.

Let $i_0=\|\{(F,a,b)\,|\,F\in N,\ a<_F b\}\|+1$ ($\|B\|$ denotes the cardinality of set B), and suppose that $r_{i_0+1}(F_0,a_0,b_0)>r_{i_0}(F_0,a_0,b_0)=r_0$ for some $a_0<_{F_0}b_0$. Then there exists $p\colon F_0\to w_0F_1\dots F_kw_k\in P$ such that $r_{i_0+1}^{(p)}(F_0,a_0,b_0)>r_0$, i.e. for some $a(F_j)\in H_n$ we have

$$a_0(F_0)<_p a(F_j)\leqq_p a'(F_m)<_{F_m} b'(F_m)<_p b_0(F_0),$$

where $x(a(F_j))>r_0$, and consequently $r_{i_0}(F_m,a',b')>r_{i_0-1}(F_m,a',b')$ for some $m\in[k]$, $a'<_{F_m}b'$. (Note that $i_0\geqq 2$, else we have nothing to prove.) Let $\alpha_0=$

$=(F_0, a_0, b_0)\equiv(F^{(0)}, a^{(0)}, b^{(0)})$, $\alpha_1=(F_m, a', b')\equiv(F^{(1)}, a^{(1)}, b^{(1)})$, and construct $\alpha_2=$ $=(F^{(2)}, a^{(2)}, b^{(2)})$, ..., α_{i_0-1} in the same way. It follows that for any $0\leqq i<j\leqq i_0-1$ there exists a cut $t\in C_{F^{(i)}}$ and a leaf u of t labelled by $F^{(j)}$ for which we have:

$$a^{(i)}(rt(t)) <_t a^{(j)}(u) <_{F^{(j)}} b^{(j)}(u) <_t b^{(i)}(rt(t)). \tag{1}$$

Moreover, by Lemma 3.2, if all the cuts t with property (1) are such that neither

$$a^{(i)}(rt(t)) <_t^R a^{(j)}(u) \text{ nor } b^{(j)}(u) <_t^R b^{(i)}(rt(t)), \tag{2}$$

then $r_{i_0-i+1}(\alpha_i)=r_{i_0-j+1}(\alpha_j)$. By the choice of i_0, $\alpha_i=\alpha_j$ for some $0\leqq i<j\leqq i_0-1$. In this case, however, $r_{i_0-i+1}(\alpha_i)=r_{i_0-j+1}(\alpha_j)$ is impossible, thus (2) contradicts the anr property. We conclude that $r_{i_0+1}(F, a, b)=r_{i_0}(F, a, b)$ for all $F\in N$, $a<_F b$, which was to be proved.

(b): $q(F, a)$ is finite.

Let $n_0=\|\{(F, a)|F\in N,\ a\in S(F)\}\|+1$, and suppose that $q_{n_0+1}(F_0, a_0)>$ $>q_{n_0}(F_0, a_0)=q_0$ for some $a_0\in S(F_0)$. Then there exists p: $F_0\to w_0F_1\ldots F_kw_k\in P$ such that $q^{(p)}_{n_0+1}(F_0, a_0)>q_0$, i.e. for some $a(F_j)\in M_n$ we have $a_0(F_0)<_p a(F_j)$ and $y(a(F_j))>q_0$. Choose this $a(F_j)$ so that

(i) $y(a(F_j))=q_{n_0}(F_j, a)$;

(ii) if $a_0(F_0)<_p a'(F_m)<_p a(F_j)$, then $y(a'(F_m))>q_{n_0}(F_m, a')$.

Clearly, $q_{n_0}(F_j, a)>q_{n_0-1}(F_j, a)$. Let $\beta_0=(F_0, a_0)\equiv(F^{(0)}, a^{(0)})$, $\beta_1=(F_j, a)\equiv$ $\equiv(F^{(1)}, a^{(1)})$, and construct $\beta_2=(F^{(2)}, a^{(2)})$, ..., β_{n_0-1} in the same way. It follows that for any $0\leqq k<m\leqq n_0-1$ there exists a cut $t\in C_{F^{(k)}}$ and a leaf u of t labelled by $F^{(m)}$ for which:

(i) $a^{(k)}(rt(t))<_t a^{(m)}(u)$;

(ii) if $a^{(k)}(rt(t))\geqq_t^R a^{(m)}(u)$, then $q_{n_0-k+1}(\beta_k)=q_{n_0-m+1}(\beta_m)$.

By the choise of n_0, $\beta_k=\beta_m$ for some $0\leqq k<m\leqq n_0-1$. As in the part (a) of the proof, $q_{n_0-k+1}(\beta_k)=q_{n_0-m+1}(\beta_m)$ is again impossible, thus (3) contradicts the anr property. Consequently, $q_{n_0+1}(F, a)=q_{n_0}(F, a)$ for all $F\in N$, $a\in S(F)$.

Now we construct $\mathscr{A}$, μ, χ, ϱ and π_0 such that $\mathscr{G}$ is amp with respect to $\mathscr{D}=(\mathscr{A}, \mu, \chi, \varrho, \pi_0)$. Let

$$\mathscr{A} = A_i\cup\{(a, i)|a\in A_s,\ 1\leqq i\leqq \max(q(F, a)|F\in N \text{ such that } a\in S(F))\}.$$

Define the mappings α: $\mathscr{A}\to A$ and β: $\mathscr{A}\to\mathbf{N}$ by

$$\alpha(c)=\begin{cases} b & \text{if } c=b\in A_i,\\ a & \text{if } c=(a, i),\ a\in A_s\end{cases} \qquad \beta(c)=\begin{cases}1 & \text{if } c\in A_i,\\ i & \text{if } c=(a, i),\ a\in A_s.\end{cases}$$

To simplify the formalism let $q(F, b)=1$ for each $b\in I(F)$. For $F\in N$ let

$$\mu(F)=\{c|\alpha(c)\in v(F) \text{ and } \beta(c)\leqq q(F, \alpha(c))\},$$

and if $c\in\mu(F)$, then

$$\chi_F(c)=\begin{cases}\{\alpha(c)\} & \text{if } \beta(c)=q(F, \alpha(c)),\\ \emptyset & \text{otherwise.}\end{cases}$$

For p: $F_0\to w_0F_1\ldots F_kw_k$ and $a_0\in S(F_0)$ consider the inherited pass-numbers $q_p(F_j, a, a_0)$ computed in step (3) of Algorithm 3.1. Extend q_p to triples (F_j, c, a_0) $c\in\mu(F_j)$ by

$$q_p(F_j, c, a_0)=q_p(F_j, \alpha(c), a_0)-(q(F_j, \alpha(c))-\beta(c)).$$

For each $c_0 \in \mu(F_0)$, if $c_0 \in \alpha^{-1}(A_s)$, then let

$$\varrho_p(c_0) = \{c(F_j) | \alpha(c_0)(F_0) <_p \alpha(c)(F_j) \quad \text{and} \quad \beta(c_0) = q_p(F_j, c, \alpha(c_0))\},$$

else let $\varrho_p(c_0) = \emptyset$. Finally, let $\pi_0 = (\mathscr{A}_1, \ldots, \mathscr{A}_m)$, where $m = \max\{q, (Z, a) | a \in \nu(Z)\}$ and for every $i \in [m]$ $\mathscr{A}_i = \{c \in \mu(Z) | \beta(c) = i\}$.

The reader is advised to construct the atomic pass description of the example *AG*. Since this is similar to that of Example 3.1, we do not detail it here.

Lemma 3.4. Let $F \in N$, $a \in S(F)$ and $t \in D_F$ be arbitrary, $\pi = (\mathscr{A}_1, \ldots, \mathscr{A}_n)$ an ordered subpartition of $\mu(F)$ such that

(i) $\bigcup_{j=1}^{n} \mathscr{A}_j = \{c \in \mu(F) | a \leqq_F \alpha(c)\}$,

(ii) $(a, q(F, a) - i) \in \mathscr{A}_{n-i}$ for every $0 \leqq i < q(F, a)$,

(iii) if $a <_F b$ and $b \in \mathscr{A}_j$, then $j \leqq n - r(F, a, b)$.

There exists an n-pass partial computation sequence for t respecting $\mathscr{D}/\pi$.

Proof. Induction on the depth of t. For $t = F(w)$ ($F \to w \in P$ for some $w \in T^*$) the statement is obvious. Let $t = F_0(w_0 t_1 \ldots t_k w_k)$ such that the top production of t is p: $F_0 \to w_0 F_1 \ldots F_k w_k$, and let $a_0 \in S(F_0)$. It is enough to prove the statement for the subpartition $\tilde{\pi}_0 = (\mathscr{A}_1^{(0)}, \ldots, \mathscr{A}_{n_0}^{(0)})$ of $\mu(F_0)$ which satisfies (i)—(iii), moreover, $n_0 = q(F_0, a_0)$ and $b_0 \in \mathscr{A}_{n_0 - r(F_0, a_0, b_0)}^{(0)}$ if $a_0 <_{F_0} b_0$. For $j \in [k]$ and $a \in S(F_j)$ such that $a_0(F_0) <_p a(F_j)$ let $\pi_j(a) = (\mathscr{A}_1^{(j)}, \ldots, \mathscr{A}_{n_j}^{(j)})$, where $n_j = q_p(F_j, a, a_0)$ and

$$\mathscr{A}_i^{(j)} = \{c(F_j) | a \leqq_{F_j} \alpha(c) \quad \text{and} \quad q_p(F_j, c, a) = i\}.$$

As $t_j \in D_{F_j}$ and $\pi_j(a)$ satisfy the requirements of the lemma, by the induction hypothesis there exist n_j-pass partial computation sequences $h_j(a)$ for t_j respecting $\mathscr{D}/\pi_j(a)$. Let

$$h_j = \text{merge}\,(h_j(a) | a \in S(F_j),\ a_0(F_0) <_p a(F_j)).$$

By Lemma 3.1, h_j respect $\mathscr{D}/\pi_j$, where $\pi_j = (\mathscr{B}_1^{(j)}, \ldots, \mathscr{B}_{n_j}^{(j)})$ is an appropriate ordered subpartition of $\mu(F_j)$. Observe that $n_j \leqq n_0$, and by the construction of ϱ we have $\mathscr{B}_i^{(j)} = \varrho_p(a_0, i) \cap \mu(F_j)$. This implies that $(\chi_{F_0}(\mathscr{A}_1^{(0)}), \ldots, \chi_{F_0}(\mathscr{A}_{n_0}^{(0)})) \circ (h_1, \ldots, h_k)$ is an n_0-pass partial computation sequence for t respecting $\mathscr{D}/\pi_0$, which was to be proved.

Proposition 3.1. If $\mathscr{G}$ is anr, then it is amp.

Proof. First suppose that $\mathscr{G}$ is connected, i.e.:

a) none of the i-attributes of any $F \in N$ is isolated in ids (F);

b) for every p: $F_0 \to w_0 F_1 \ldots F_k w_k$ and $a \in S(F_j)$ ($j \in [k]$) there exists $a_0 \in S(F_0)$ such that $a_0(F_0) <_p a(F_j)$.

By Lemmas 3.1 and 3.4 every $t \in D_Z$ has a (total) computation sequence respecting $\mathscr{D}/\pi_0$, thus $\mathscr{G}$ is atomic m-pass with respect to $\mathscr{D}$. To treat the general case extend $\mathscr{G}$ as follows. For each $F \in N$ let $\bar{\nu}(F) = \nu(F) \cup \{\bar{a}\}$, where $\bar{a}$ is a new s-attribute. Correspondingly, for each p: $F_0 \to w_0 F_1 \ldots F_k w_k$ add the rule $\bar{a}(F_0) = \bar{f}(b_1(F_0), \ldots, b_i(F_0), a_1(F_{j_1}), \ldots, a_n(F_{j_n}), \bar{a}(F_1), \ldots, \bar{a}(F_k))$ to r_p, where $b_1, \ldots, b_i$ are all the isolated i-attributes of ids (F_0), $a_1(F_{j_1}), \ldots, a_n(F_{j_n})$ are all those s-attributes

that cannot be connected with any $a_0 \in S(F_0)$ in $\mathrm{idp}(p)$ and $\bar{f}$ is a hypothetical function. Let $\bar{\mathcal{G}}$ denote this extended grammar. As $\bar{\mathcal{G}}$ is connected, we can construct $\bar{\mathcal{A}}$, $\bar{\mu}\bar{\chi}$, $\bar{\varrho}$ and $\bar{\pi}_0$ as above. Returning to the original grammar $\mathcal{G}$, let $\mathcal{A}=\bar{\mathcal{A}}$, $\mu(F)=\bar{\mu}(F)$ for each $F \in N$,

$$\chi_F(c) = \begin{cases} \bar{\chi}_F(c) & \text{if } \alpha(c) \neq \bar{a}, \\ \emptyset & \text{if } \alpha(c) = \bar{a}, \end{cases}$$

$\varrho=\bar{\varrho}$ and $\pi_0=\bar{\pi}_0$. It is clear that $\mathcal{G}$ is amp with respect to $\mathcal{D}=(\mathcal{A}, \mu, \chi, \varrho, \pi_0)$ defined in this way.

A *top-down assignment* of partitions for $\mathcal{G}$ is a quadruple $\mathcal{T}=(\mathcal{Q}, \tau, \varphi, q_0)$, where

(i) $\mathcal{Q}$ is a finite nonempty set, the set of states;

(ii) $\tau=\{\tau_F | F \in N\}$ such that τ_F is a mapping of $\mathcal{Q}$ into the set Π_F of all ordered partitions of $v(F)$;

(iii) $\varphi=\{\varphi_p | p \in P\}$ such that if $p\colon F_0 \to w_0 F_1 \dots F_k w_k$, then φ_p is a partial mapping of $\mathcal{Q}$ into $\mathcal{Q}^k$.

(iv) $q_0 \in \mathcal{Q}$ is the initial state.

Concerning states, $\mathcal{T}$ is considered as an ordinary deterministic top-down tree automaton working on the derivation trees of G. $s_{qt}(u)$ will denote the state in which $\mathcal{T}$ passes through a node u of a derivation tree t, starting from state q at the root. q will be omitted if $t \in D_Z$ and $q=q_0$.

Let t be a derivation tree and $q \in \mathcal{Q}$. An m-pass computation sequence h for t is said to respect $\mathcal{T}/q$ if $E_h(u)=\tau_F(s_{qt}(u))$ for all nodes $u(\in U_t(F))$.

Definition 3.3. $\mathcal{G}$ is generalized uniform m-pass with respect to $\mathcal{T}$ if every complete derivation tree has an m-pass computation sequence respecting $\mathcal{T}/q_0$. $\mathcal{G}$ is generalized uniform multi-pass (gump) if there exist $m \in \mathbf{N}$ and $\mathcal{T}$ such that $\mathcal{G}$ is generalized uniform m-pass with respect to $\mathcal{T}$. $\mathcal{G}$ is uniform multi-pass (ump) if $\mathcal{T}$ can be chosen so that $\mathcal{Q}=\bigcup(\Pi_F | F \in N)$, $\tau_F(\pi)=\pi$ for all $F \in N$, $\pi \in \Pi_F$, and φ is a top-down assignment of partitions in the sense of [5].

Lemma 3.5. If $\mathcal{G}$ is gump, then it is ump, too.

Proof. Let $\mathcal{G}$ be gump with respect to $\mathcal{T}$. For each $F \in N$ let

$$\mathcal{Q}_F = \{s_t(u) | t \in D_Z,\ u \in U_t(F)\}$$

be the set of possible states at an F-labelled node. Consider an arbitrary $t \in D_Z$ and a node $u \in U_t(F)$. Intervene in the work of $\mathcal{T}$ on t at node u as follows.

(i) Force $\mathcal{T}$ to change the state $s_t(u)$ to any other state $q \in \mathcal{Q}_F$ for which $\tau_F(q)=\tau_F(s_t(u))$.

(ii) Let it continue working as if q were the state in which it had reached u.

Let $s'(v)$ denote the new state assigned to any $v \in U_t$ in the above way. By Lemma 2.1 there exists a computation sequence h' for t such that $E_{h'}(v)=\tau_Y(s'(v))$ for all nodes $v \in U_t(Y)$. By successive interventions $\mathcal{T}$ can be forced to perform a uniform top-down assignment on t, thus $\mathcal{G}$ can be made uniform. Moreover, the pass-number m also remains the same.

Proposition 3.2. If $\mathcal{G}$ is amp, then it is ump.

Proof. Let $\mathcal{G}$ be atomic m-pass with respect to $\mathcal{D}=(\mathcal{A}, \mu, \chi, \varrho, \pi_0)$. Define $\mathcal{T}=(\mathcal{Q}, \tau, \varphi, q_0)$ as follows. $\mathcal{Q}=\bigcup(\{F\}\times\Pi_{\mu(F)}|F\in N)$, where $\Pi_{\mu(F)}$ denotes the set of all ordered m-partitions of $\mu(F)$; $q_0=(Z, \pi_0)$. For each $F\in N$ let

$$\tau_F(F, (\mathcal{A}_1, \ldots, \mathcal{A}_m)) = (\chi_F(\mathcal{A}_1), \ldots, \chi_F(\mathcal{A}_m)).$$

If p: $F_0 \to w_0 F_1 \ldots F_k w_k \in P$, then

$$\varphi_p(F_0, \pi^{(0)}) = ((F_1, \pi^{(1)}), \ldots, (F_k, \pi^{(k)})),$$

where $\pi^{(n)}=(\mathcal{A}_1^{(n)}, \ldots, \mathcal{A}_m^{(n)})$ $0\leqq n\leqq k$, and for any $j\in[k]$ and $c\in\mu(F_j)$, $c\in\mathcal{A}_i^{(j)}$ iff i is the minimal number such that $c(F_j)\in\varrho_p(\mathcal{A}_i^{(0)})$. Let h be an m-pass computation sequence for $t\in D_F$ and $\pi\in\Pi_{\mu(F)}$. An easy induction on the depth of t shows that h respects $\mathcal{D}/\pi$ iff it respects $\mathcal{T}/(F, \pi)$. Thus, $\mathcal{G}$ is gump with respect to $\mathcal{T}$, and by Lemma 3.5 it is ump, too.

Proposition 3.3. If $\mathcal{G}$ is ump, then it is anr.

Proof. Let $t\in D_Z$, $u\in U_t(F)$ and $a<_F b$ be arbitrary. It was proved in [5] (Lemma 4) that if h is a uniform computation sequence for t, then $b(u)$ must be evaluated sooner than $a(u)$ in h. Suppose $\mathcal{G}$ were not anr. It is clearly anc, so there must be a cut $t\in C_F$, a leaf $u\in U_t(F)$ and an attribute $a\in v(F)$ such that $a(rt(t))<_t^R a(u)$, or vice versa. Let t^k denote the cut which can be obtained by k-fold composition of t at node u. Since $\mathcal{G}$ is reduced, there exists a complete derivation tree t_0 that contains t^k. But in idt (t_0) there is a path containing at least k R-arcs, thus $\mathcal{G}$ cannot be uniform m-pass for any $m\leqq k$. This is a contradiction, since k is arbitrary.

By Propositions 3.1, 3.2 and 3.3 we have proved

Theorem 3.1. $\mathcal{G}$ is anr iff it is amp iff it is ump.

Let $\mathcal{G}$ be uniform m-pass with respect to τ. If $\pi=(A_1, \ldots, A_m)\in\mathcal{Q}$ and $a\in\bigcup_{i=1}^{m} A_i$, then let $i_\pi(a)$ denote the number i for which $a\in A_i$.

Lemma 3.6. (a) For every $F\in N$, $a<_F b$ and $\pi\in\mathcal{Q}_F$:

$$i_\pi(a)-i_\pi(b) \geqq r(F, a, b).$$

(b) For every $F\in N$, $a\in S(F)$ and $\pi\in\mathcal{Q}_F$, $i_\pi(a)\geqq q(F, a)$.

Proof. (a) Let $\bar{r}(F, a, b)=\min(i_\pi(a)-i_\pi(b)|\pi\in\mathcal{Q}_F)$. We have to prove that $\bar{r}(F, a, b)\geqq r(F, a, b)$. Fix $F_0\in N$, $a_0<_{F_0} b_0$ and p: $F_0 \to w_0 F_1 \ldots F_k w_k$ arbitrarily, and let $\pi_0\in\mathcal{Q}_{F_0}$. If $\varrho_p(\pi_0)=(\pi_1, \ldots, \pi_k)$, then there exists a complete derivation tree t and nodes $u_0, u_1, \ldots, u_k$ of t (u_n labelled by F_n) representing an occurrence of p such that $s_t(u_n)=\pi_n$ for every $0\leqq n\leqq k$. Let

$$a_0(F_0) <_p a_1(F_{i_1}) <_{F_{i_1}} b_1(F_{i_1}) <_p \ldots <_p a_n(F_{i_n}) <_{F_{i_n}} b_n(F_{i_n}) <_p b_0(F_0)$$

be any path in idp (p) containing l R-edges. Since there exists an m-pass computation sequence for t respecting τ/q_0,

$$i_{\pi_0}(a_0)-i_{\pi_0}(b_0) \geqq l+\sum_{j=1}^{n}(i_{\pi_{i_j}}(a_j)-i_{\pi_{i_j}}(b_j)) \geqq l+\sum_{j=1}^{n}\bar{r}(F_{i_j}, a_j, b_j).$$

Thus, $\bar{r}(F_0, a_0, b_0) \geqq l + \sum_{j=1}^{n} \bar{r}(F_{i_j}, a_j, b_j)$. However, by construction $r(F_0, a_0, b_0)$ is the minimal number satisfying this property, i.e. $r(F_0, a_0, b_0) \leqq \bar{r}(F_0, a_0, b_0)$ for all $F_0 \in N$, $a_0 <_{F_0} b_0$. Statement (b) can be proved similarly.

Theorem 3.2. If $\mathcal{G}$ is anr, then $m_0 = \max(q(Z, a) | a \in v(Z))$ is the minimal number m for which $\mathcal{G}$ is atomic m-pass.

Proof. Suppose $\mathcal{G}$ were atomic m-pass for some $m < m_0$. Then, by Proposition 3.2 $\mathcal{G}$ is uniform m-pass, too, which contradicts (b) of Lemma 3.6.

Now we give an algorithm that decides the anr property. It is assumed that the well-known test for the anc property has been executed before. In the first step we construct the set

$$R = \{(F, a, b) | F \in N, \text{ there exists } t \in D_F \text{ such that } a(rt(t)) <_t^R b(rt(t))\}.$$

Step (2) constructs for each $F \in N$ and $a \in S(F)$ the set

$$R(a(F)) = \{a'(Y) | a' \in S(Y) \text{ and there is a cut } t \in C_F \text{ and a leaf } u \in U_t(Y) \text{ such that } a(rt(t)) <_t^R a'(u)\},$$

while step (3) constructs for each $F \in N$ and $b \in I(F)$ the set

$$R(b(F)) = \{b'(Y) | b' \in I(Y) \text{ and there is a cut } t \in C_Y \text{ and a leaf } u \in U_t(F) \text{ such that } b(u) <_t^R b'(rt(t))\}.$$

Algorithm 3.2. *Input:* An anc *AG* $\mathcal{G}$.

Output: "yes" if $\mathcal{G}$ is anr, "no" otherwise.

(1) Construct the set $R \subseteq N \times A_s \times A_i$ as follows.

(a) Let $R_0 = \emptyset$ and set $i = 0$.

(b) For each p: $F_0 \to w_0 F_1 \dots F_k w_k \in P$ let

$$R_{i+1}^{(p)} = \{(F_0, a_0, b_0) | a_0(F_0) <_p^R b_0(F_0) \quad \text{or} \quad a_0(F_0) <_p a(F_j) <_{F_j} b(F_j) <_p b_0(F_0) \text{ for some } (F_j, a, b) \in R_i\}.$$

Let $R_{i+1} = R_i \cup (\cup(R_{i+1}^{(p)} | p \in P))$. If $R_{i+1} = R_i$, then let $R = R_i$, else set $i = i+1$ and repeat step (b).

(2) Construct for each $F \in N$, $a \in S(F)$ the set $R(a(F))$ as follows.

(c) Let $L_0(a(F)) = \{a(F)\}$, $R_0(a(F)) = \emptyset$ and set $i = 0$.

(d) For each p: $F_0 \to w_0 F_1 \dots F_k w_k \in P$ and $a_0 \in S(F_0)$ let

$$L_{i+1}^{(p)}(a_0(F_0)) = \cup(L_i(a(F_j)) | j \in [k], \ a_0(F_0) <_p a(F_j));$$

$$R_{i+1}^{(p)}(a_0(F_0)) = \cup(R_i(a(F_j)) | j \in [k], \ a_0(F_0) <_p a(F_j)) \cup$$

$$\cup(L_i(a(F_j)) | j \in [k], \ a_0(F_0) <_p^R a(F_j), \quad \text{or} \quad a_0(F_0) <_p a'(F_m) <_{F_m} <_{F_m} b'(F_m) <_p a(F_j) \quad \text{and} \quad (F_m, a', b') \in R).$$

Let $L_{i+1}(a(F)) = L_i(a(F)) \cup (\cup(L_{i+1}^{(p)}(a(F)) | p \in P))$;

$$R_{i+1}(a(F)) = R_i(a(F)) \cup (\cup(R_{i+1}^{(p)}(a(F)) | p \in P)).$$

If both $L_{i+1}(a(F)) = L_i(a(F))$ and $R_{i+1}(a(F)) = R_i(a(F))$ for all $F \in N$, $a \in S(F)$, then let $R(a(F)) = R_i(a(F))$, else set $i = i+1$ and repeat step (d).

(3) Construct for each $F\in N$, $b\in I(F)$ the set $R(b(F))$ as follows.
(e) Let $L_0(b(F))=\{b(F)\}$, $R_0(b(F))=\emptyset$ and set $i=0$.

(f) For each $p\colon F_0\to w_0F_1\dots F_kw_k\in P$, $F\in N$ and $b\in I(F)$ let

$$L_{i+1}^{(p)}(b(F))=\cup(L_i(b_0(F_0))|F=F_j \text{ for some } j\in[k] \text{ and } b(F_j)<_p b_0(R_0))$$

$$R_{i+1}^{(p)}(b(F))=\cup(R_i(b_0(F_0))|F=F_j \text{ and } b(F_j)<_p b_0(F_0))\cup$$

$$\cup(L_i(b_0(F_0))|F=F_j \text{ and } b(F_j)<_p^R b_0(F_0))\cup$$

$$\cup(L_i(b_0(F_0))|F=F_j \text{ and } b(F_j)<_p a'(F_m)<_{F_m} b'(F_m)<_p b_0(F_0) \text{ for some } m\in[k],\ (F_m,a',b')\in R).$$

Let $L_{i+1}(b(F))=L_i(b(F))\cup(\cup(L_{i+1}^{(p)}(b(F))|p\in P))$;

$$R_{i+1}(b(F))=R_i(b(F))\cup(\cup(R_{i+1}^{(p)}(b(F))|p\in P)).$$

If both $L_{i+1}(b(F))=L_i(b(F))$ and $R_{i+1}(b(F))=R_i(b(F))$ for all $F\in N$, $b\in I(F)$, then let $R(b(F))=R_i(b(F))$, else set $i=i+1$ and repeat step (f).

(4) Output "no" if there exist $F\in N$ and $a\in v(F)$ such that $a(F)\in R(a(F))$. Otherwise output "yes". Halt.

Theorem 3.3. The atomic (uniform) m-pass property can be decided in polynomial time.

Proof. By Theorem 3.2 it is enough to prove that Algorithms 3.1 and 3.2 are both polynomial. The size of $\mathcal{G}$ will be expressed in the parameters $n=\|N\|$, $p=\|P\|$, $a=\|A\|$ and x, which is the maximum number of nonterminal occurrences in a production. First consider Algorithm 3.2. One execution of step (b) requires $O(pxa^2)$ time, and it must be executed at most $\|R\|<na^2$ times. Thus, the total amount of time required for step (1) is $O(pxna^4)$. In step (2) consider the sum

$$S_i=\sum(\|L_i(a(F))\|+\|R_i(a(F))\|\ |F\in N,\ a\in S(F)) \quad \text{for each } i.$$

Since $2na$ is an upper bound for all the S_i, step (d) must be executed at most $O(na)$ times. One execution of step (d) requires $O(pxa^2)$ time, so step (2), as well as step (3), needs $O(pxna^3)$ amount of time. Thus, the complexity of Algorithm 3.2 is $O(pxna^4)$. Now consider Algorithm 3.1. One execution of step (b) requires $O(pxa^2)$ time, and by Lemma 3.3 it must be executed at most $i_0<na$ times. Similarly, step (d) must be executed at most $n_0<na$ times, and one execution needs $O(pxa^2)$ time. Step (3) is not relevant, thus the complexity of Algorithm 3.1 is $O(pxna^4)$, too.

4. Implementation

As it is usual, a node u of a derivation tree is considered an object consisting of the following data.

— $u.prod$: number of production applied at u;
— $u.j$: reference to the j-th son of u;
— $u.a$: attribute a at u for each $a\in v(u)$;
— $u.on\_c$ and $u.off\_c$: Boolean flags for each atom $c\in\mu(u)$.

The initial value of all the flags is false. *u.on_c* can be set to true by procedure *activate* (u, c), while procedure *release* (u, c) sets *u.on_c=false* and *u.off_c= =true*. Boolean functions *active* (u, c) and *done* (u, c) are used to test the value of *on_c* and *off_c* at u, respectively. For each $F \in N$ we have a procedure *F_pass* (u), which makes one visit to an F-labelled node u as follows.

Procedure *F_pass* (u) *node u*
begin
for each $c \in \mu(F)$
if *active* (u, c) then
begin
case *u.prod* of
⋮
p: comment p: $F_0 \to w_0 F_1 \ldots F_k w_k$
for $j=1$ to k
begin
for each $d \in \varrho_p(c) \cap \mu(F_j)$
if not *done* $(u.j, d)$ then
begin
activate $(u.j, d)$;
evaluate attributes $\chi_{F_j}(d) \cap I(F_j)$ at $u.j$
end;
F_j_pass $(u.j)$
end;
evaluate attributes $\chi_{F_0}(c) \cap S(F_0)$ at u
⋮
esac;
release (u, c)
end
end

Procedure *evaluate* (u) controls the evaluation of a complete derivation tree with root u.

Procedure *evaluate* (u) *node u*
comment $\pi_0 = (\mathscr{A}_1, \ldots, \mathscr{A}_m)$
begin
activate (c, u) for all $c \in \mathscr{A}_1$;
⋮
Z_pass (u);
⋮
activate (c, u) for all $c \in \mathscr{A}_m$;
Z_pass (u)
end

5. Conclusion

The main advantage of the atomic characterization is that the evaluation procedure can be implemented in a simple and efficient way. It is not practial to base the evaluation procedure directly on a uniform top-down assignment of partitions, since the number of possible partitions is exponential. Using the atomic characteriza-

tion, however, the size of implementation becomes polynomial, and the characterization itself can be carried out in polynomial time. Unfortunately, the atomic characterization of uniform multi-visit (i.e. anc) AG is not so useful (although it is possible to do it) because it can be proved that the decision problem of the uniform m-visit property is P-space complete. On the other hand, if we consider subclasses of anc AG in which the way of walking through the derivation trees is fixed independently of the grammar, then the atomic characterization is always useful. For example, it is worth doing the atomic characterization of ASE-like anc AG in which the attributes are evaluated in alternative left-to-right and right-to-left passes.

Abstract

A natural characterization of uniform multi-pass attribute grammars is introduced. An easy algorithm is given to test the uniform multi-pass property, and it is proved that the uniform m-pass property for fixed $m \in \mathbf{N}$ is decidable in polynomial time.

RESEARCH GROUP ON THEORY OF AUTOMATA
HUNGARIAN ACADEMY OF SCIENCES
SOMOGYI U. 7.
SZEGED, HUNGARY
H—6720

DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI V. TERE 1.
SZEGED, HUNGARY
H—6720

References

[1] ENGELFRIET, J. and G. FILE, Simple multi-visit attribute grammars, Journal of Computer and System Sciences, v. 24, 1982. pp. 283—314.
[2] ENGELFRIET, J. and G. FILE, Passes and paths of attribute grammars, Information and Control, v. 49, 1981, pp. 125—169.
[3] KENNEDY, K. and S. K. WARREN, Automatic generation of efficient evaluators for attribute grammars, Conf. Record of the Third ACM Symp. on Principles of Programming Languages, 1976, pp. 32—49.
[4] KNUTH, D. E., Semantics of context free languages, Math. Systems Theory, v. 2, 1968, pp. 127—145.
[5] NIELSON, H. R., Computation sequences: a way to characterize classes of attribute grammars, Acta Informatica, v. 19, 1983, pp. 255—268.

Received July 14, 1984

On the equivalence of the frontier-to-root tree transducers I.

By Z. ZACHAR

It is known in finite automata theory that the equivalence problem can be traced back to the isomorphism of automata. Then, in a natural way, one can raise the question whether two frontier-to-root tree transducers (F-transducers) are isomorphic if they are equivalent.

In this paper we deal with this problem. We introduce the class of the connected F-transducers with adapted rules and that of the inferior F-transducers. It will be shown that for each F-transducer there are equivalent F-transducers from the above classes.

Moreover, in the second part we define a subclass of the class of deterministic F-transducers, namely the class of normalized F-transducers. It will be proved that two strongly normalized F-transducers are equivalent if and only if they are isomorphic.

The terminology is used in the sense of [1]. The algebraic notations developed by Gécseg and Steinby in [3, 4] will be used throughout this paper.

1. Notions and notations

By a *ranked alphabet* we mean a finite nonvoid union $F=\cup(F_k|k=0, 1, \ldots)$ of pairwise disjoint sets F_k.

Take an arbitrary ranked alphabet F and a set R. Then the set of all *F-trees* over R (or trees, for short) is the smallest set $T_F(R)$ satisfying the following conditions.

(i) $F_0 \cup R \subseteq T_F(R)$.

(ii) If $f \in F_k$ $(k>0)$ and $p_1, \ldots, p_k \in T_F(R)$ then $f(p_1, \ldots, p_k) \in T_F(R)$.

We can define the *height* $(h^S(p))$ and *frontier* $(fr^S(p))$ of a tree $p(\in T_F(R))$ with respect to $S(\subseteq R)$ in the following way:

(i) if $p \in T_F(R \setminus S)$ then $fr^S(p)=\varepsilon$, $h^S(p)$ is undefined,

(ii) if $p \in S$ then $fr^S(p)=p$, $h^S(p)=0$, and

(iii) if $p=f(p_1, \ldots, p_k)(\in T_F(R) \setminus T_F(R \setminus S))$ then $fr(p)=fr(p_1) \ldots fr(p_k)$ and $h^S(p)=\max(h^S(p_i)|i=1, \ldots, k)+1$.

Here ε denotes the empty string. If $S=R$ then the symbol S can be omitted.

The set of *subtrees* $(\mathrm{sub}(p))$ and the set of proper subtrees $(sub(p))$ of a tree p are defined in the usual way.

In the rest of this paper the pairwise disjoint sets of variables $X=\{x_1, x_2, \ldots\}$, $Y=\{y_1, y_2, \ldots\}$ and $Z=\{z_1, z_2, \ldots\}$ are kept fixed. The symbols $z_1, z_2, \ldots$ are used as auxiliary variables. For arbitrary integer $n(\geqq 0)$, X_n, Y_n and Z_n denote the sets $\{x_1, \ldots, x_n\}$, $\{y_1, \ldots, y_n\}$ and $\{z_1, \ldots, z_n\}$, respectively.

If $p\in T_F(X_n\cup Z_k)$ and $fr^Z(p)=z_{i_1}\ldots z_{i_l}$ then for p we also use the notations $p(z_1, \ldots, z_k)$ and $p\langle z_{i_1}, \ldots, z_{i_l}\rangle$. Substituting $t_i(\in T_F(X_n\cup Z))$ $(1\leqq i\leqq k)$ for the auxiliary variable z_i $(1\leqq i\leqq k)$ in a tree p we obtain another tree which is denoted by $p(t_1, \ldots, t_k)$. Let $p=q(z_{i_1}, \ldots, z_{i_l})$ where $q\in T_F(X_n\cup Z_l)$ and $fr^Z(q)=z_1\ldots z_l$. Then $p\langle t_1, \ldots, t_l\rangle$ will stand for $q(t_1, \ldots, t_l)$ $(t_i\in T_F(X_n\cup Z)$, $i=1, \ldots, l)$, that is the tree $p\langle t_1, \ldots, t_l\rangle$ is obtained by replacing each variables of $z_{i_1}, \ldots, z_{i_l}$ by the tree $t_1, \ldots, t_l$ one after another.

The auxiliary variable z_1 of Z_1 will also be denoted by $\#$.

In the sequel we shall use the notations

$\hat{T}_F(X_n) = \{p | p\in T_F(X_n\cup Z_1), fr^{Z_1}(p) = \#\}$ and

$\tilde{T}_F(X_n) = T_F(X_n\cup Z_1)\setminus T_F(X_n)$.

If $\bar{p}\in\tilde{T}_F(X_n)$ and $p\in T_F(X_n)$ then we denote the tree $\bar{p}(p)$ by $p\cdot\bar{p}$.

Now we can define the set of the *supertrees* $(\sup(p))$ for a tree $p(\in T_F(X_n))$: $\bar{q}\in\tilde{T}_F(X_n)$ is in $\sup(p)$ if there exists a $q\in T_F(X_n)$ such that $p=q\cdot\bar{q}$.

We now turn to the definition of a frontier-to-root tree transducer (F-transducer). An *F-transducer* is a system $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$, where F and G are ranked alphabets, A is a finite nonvoid set of *states*, $A'\subseteq A$ is the set of *final states*, and Σ is a finite set of rewriting *rules* of the following two types:

(i) $x\to aq$ $(x\in X_n\cup F_0,\ a\in A,\ q\in T_G(Y_m))$ and

(ii) $f(a_1, \ldots, a_k)\to aq(z_1, \ldots, z_k)$ $(f\in F_k,\ k>0,\ a_1, \ldots, a_k,\ a\in A,\ q\in T_G(Y_m\cup Z_k))$.

The transformation induced by $\mathbf{A}$ will be denoted by $\tau_{\mathbf{A}}$. Moreover, let $\mathrm{dom}\,\tau_{\mathbf{A}}$ and $\mathrm{range}\,\tau_{\mathbf{A}}$ be, respectively, the domain and range of $\tau_{\mathbf{A}}$. For an arbitrary tree p we put $\tau_{\mathbf{A}}(p)=\{q|(p, q)\in\tau_{\mathbf{A}}\}$.

For an F-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ and two sets $A_1, A_2(\subseteq A)$ we denote by $\tau^{A_1}_{\mathbf{A},A_2}$ the transformation induced by

$$(T_F(X_n\cup Z_1), A, T_G(Y_m\cup Z_1), A_1, \Sigma\cup\{\#\to a\#\,|a\in A_2\}).$$

Moreover, let

$dom\,\tau^{A_1}_{\mathbf{A},A_2} = \mathrm{dom}\,\tau^{A_1}_{\mathbf{A},A_2}\cap\hat{T}_F(X_n)$ and

$range\,\tau^{A_1}_{\mathbf{A},A_2} = \{q|p\in dom\,\tau^{A_1}_{\mathbf{A},A_2}, q\in\tau^{A_1}_{\mathbf{A},A_2}(p)\}$.

If $A_1=A'$ and $A_2=\emptyset$, then A_1 and A_2 will generally be omitted in $\tau^{A_1}_{\mathbf{A},A_2}$. Furthermore, if there is no danger of confusion then we write τ instead of $\tau_{\mathbf{A}}$. Let us note that a singleton will also be denoted by its element.

Take an arbitrary F-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$. If $\tau_{\mathbf{A}}$ is a partial mapping then $\mathbf{A}$ is called *functional*. Moreover, $\mathbf{A}$ is *deterministic*, if all its different rules have different left sides.

Let $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma_A)$ and $\mathbf{B}=(T_F(X_n), B, T_G(Y_m), B', \Sigma_B)$ be two F-transducers and take a bijective mapping μ of A onto B. If the following three conditions are satisfied then μ is called an *isomorphism*.

(i) $x \to aq \in \Sigma_A$ $(x \in X_n \cup F_0, a \in A)$ if and only if $x \to \mu(a)q \in \Sigma_B$.
(ii) $f(a_1, \ldots, a_k) \to a_0 q \in \Sigma_A$ if and only if $f(\mu(a_1), \ldots, \mu(a_k)) \to \mu(a_0)q \in \Sigma_B$, where $f \in F_k$ $(k>0)$ and $a_i \in A$ $(i=0, 1, \ldots, k)$.
(iii) $\mu(A')=B'$.

We can say that **A** and **B** are *isomorphic*.

Finally, two F-transducers are called *equivalent* if the transformations induced by them coincide.

2. Inferior F-transducers

Let $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ be an arbitrary F-transducer. It is called *connected*, if for each rule of the form $x \to aq$ $(x \in X_n \cup F_0)$ and $f(a_1, \ldots, a_k) \to aq$ $(f \in F_k, k>0)$ in Σ, there are trees $p_1, \ldots, p_k$, $\bar{p}$ such that $\bar{p} \in dom\, \tau_a$ and $p_i \in \mathrm{dom}\, \tau^{a_i}$ $(i=1, \ldots, k)$, moreover, the set A of states coincides with $\{a \mid p \to aq \in \Sigma\}$.

One can easily show that for every **A** there is a connected F-transducer **B** with $\tau_{\mathbf{A}}=\tau_{\mathbf{B}}$.

Definition 1. By a connected F-transducer with its *adapted rules* (AF-transducer), we mean a connected F-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ such that each state $a(\in A)$ satisfies the following conditions:

(i) if range τ^a is a singleton then for each tree $\bar{p} \in \mathrm{dom}\, \tau_a \setminus \{\#\}$, the inclusion $\tau_a(\bar{p}) \subseteq T_G(Y_m)$ holds,
(ii) if range $\tau_a \subseteq T_G(Y_m)$ then range $\tau^a=\{y_1\}$.

It is easy to prove that *range* $\tau_a \subseteq T_G(Y_m)$ if and only if range $\tau_a \subseteq T_G(Y_m)$. Thus the condition (ii) of the above definition can be replaced by the following:

(ii)′ if *range* $\tau_a \subseteq T_G(Y_m)$ then range $\tau^a=\{y_1\}$.

Lemma 2. For any connected F-transducer an equivalent AF-transducer can be constructed.

Proof. Let $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ be an arbitrary connected F-transducer. We shall construct the F-transducer $\bar{\mathbf{A}}=(T_F(X_n), A, T_G(Y_m), A', \bar{\Sigma})$ by rewriting the rules of Σ.

Assume that range $\tau^a_{\mathbf{A}}$ is a singleton i.e., for each tree $p \in \mathrm{dom}\, \tau^a_{\mathbf{A}}$, $\tau^a_{\mathbf{A}}(p)=q$. Then we replace every rule $f(a_1, \ldots, a_k) \to a_0 r$ in Σ by the rule $f(a_1, \ldots, a_k) \to a_0 r(t_1, \ldots, t_k)$, where $t_i=q$ if $a_i=a$ and $t_i=z_i$ otherwise $(i=1, \ldots, k)$.

If range $\tau_{\mathbf{A},a} \subseteq T_G(Y_m)$ then $a \notin A'$, thus every rule of the form $f(a_1, \ldots, a_k) \to ar$ and $x \to ar$ may be replaced by the rule of the form $f(a_1, \ldots, a_k) \to ay_1$ and $x \to ay_1$, resp.

It is clear that the set $\bar{\Sigma}$ of rules constructed in this way satisfies the conditions of Lemma 2.

Lemma 3. If the AF-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ is functional then for each state $a(\in A)$, τ^a and τ_a are mappings.

Proof. Assume that τ^a $(a \in A)$ is not a mapping. Then $a \notin A'$ and there are trees $p \in \mathrm{dom}\, \tau^a$ and $q_1, q_2 \in \tau^a(p)$ such that $q_1 \neq q_2$. Since range τ^a is not a singleton, thus by condition (ii) of Definition 1 there exist trees $\bar{p} \in \mathrm{dom}\, \tau_a$ and $\bar{q} \in \tau_a(\bar{p})$ such that the tree $\bar{q}$ contains the symbol $\#$ in its frontier. Then $p \cdot \bar{p} \in \mathrm{dom}\, \tau$, so

$q_i \cdot \bar{q} \in \tau(p \cdot \bar{p})$ $(i=1, 2)$. It means that $q_1 \cdot \bar{q} = q_2 \cdot \bar{q}$, therefore $q_1 = q_2$ which contradicts our assumption.

Next let us consider the transformation τ_a. We have that $\operatorname{dom} \tau_a = \operatorname{dom} \tau \cup \cup \{p | p \in \tilde{T}_F(X_n) \cap \operatorname{dom} \tau_a\}$. Since τ is a mapping, it suffices to prove that if $\bar{p} \in \operatorname{dom} \tau_a \setminus T_F(X_n) \setminus \{\#\}$ and $\bar{q}_1, \bar{q}_2 \in \tau_a(\bar{p})$ then $\bar{q}_1 = \bar{q}_2$.

If range τ^a is a singleton then by condition (i) in the definition of an AF-transducer we know that $\bar{q}_1, \bar{q}_2 \in T_G(Y_m)$. It means that for an arbitrary tree $p \in \operatorname{dom} \tau^a$ the equalities $\tau(p \cdot \bar{p}) = q_1$ and $\tau(p \cdot \bar{p}) = q_2$ hold. Consequently $\bar{q}_1 = \bar{q}_2$.

If range τ^a is not a singleton then there are trees $p_1, p_2 \in \operatorname{dom} \tau^a$ for which $\tau^a(p_1) = q_1 \neq q_2 = \tau^a(p_2)$. We have that

$$\tau(p_1 \cdot \bar{p}) = q_1 \cdot \bar{q}_1 = q_1 \cdot \bar{q}_2$$

and

$$\tau(p_2 \cdot \bar{p}) = q_2 \cdot \bar{q}_1 = q_2 \cdot \bar{q}_2,$$

which imply that $\bar{q}_1 = \bar{q}_2$. This ends the proof of Lemma 3.

Definition 4. Let $\mathbf{A} = (T_F(X_n), A, T_G(Y_m), A', \Sigma)$ be an AF-transducer. The transformation induced by the state $a (\in A)$ can *be cut* by the tree $q_a \in \tilde{T}_G(Y_m) \setminus \{\#\}$, if for all $\bar{a} \in A'$, $p \in \operatorname{dom} \tau^a_{\bar{a}}$ and $q \in \tau^a_{\bar{a}}(p)$ there is a tree $\tilde{q}$ such that $q = \tilde{q} \cdot q_a$. The tree q_a cuts the transformation τ^a maximally, if τ^a can not be cut by any tree $\bar{q} \cdot q_a$, where $\bar{q} \in \tilde{T}_G(Y_m) \setminus \{\#\}$.

By the above definition the transformation τ^a can be cut by the tree q_a if and only if q_a is a supertree of each tree from the set $\{q | q \in \text{range } \tau^a_{\bar{a}},\ \bar{a} \in A'\}$.

Theorem 5. There is an algorithm to decide for each AF-transducer $\mathbf{A} = (T_F(X_n), A, T_G(Y_m), A', \Sigma)$ and arbitrary state $a (\in A)$ whether the transformation τ^a can be cut. Moreover, every tree q_a cutting τ^a can be given effectively.

Proof. Let $K = \max (q | q \in \tau^a_{\bar{a}}(p),\ p \in \operatorname{dom} \tau^a_{\bar{a}},\ h(p) \leqq \|A\|,\ a \in A,\ \bar{a} \in A')$ and $L = (K+6) \cdot \|A\|$. We denote by Q the set $\{p | p \in \tilde{T}_F(X_n),\ h(p) \leqq L\}$. Let $a \in A$ and $q_a \in \tilde{T}_G(Y_m) \setminus \{\#\}$ be arbitrary. It is sufficient to show that the following statement is valid:

if for all $\bar{a} \in A'$, $p \in \operatorname{dom} \tau^a_{\bar{a}} \cap Q$ and $q \in \tau^a_{\bar{a}}(p)$ there exists a tree $\tilde{q}$ such that $q = \tilde{q} \cdot q_a$, then the transformation τ^a can be cut by q_a i.e., for all $\bar{a} \in A'$, $p \in \operatorname{dom} \tau^a_{\bar{a}}$ and $q \in \tau^a_{\bar{a}}(p)$ the tree q_a is a supertree of q. Obviously, every such q_a can be given effectively.

The proof of this statement can be performed by induction. If $h(p) \leqq L$ then by our assumption the tree q_a is a supertree of each tree from the sets range $\tau^a_{\bar{a}}$ $(\bar{a} \in A')$. Now let $h(p) > L$ and assume that our statement holds for all trees which have less number of occurrences of symbols from F than p has. Then there are two sequences $p_0, \ldots, p_{K+6}$ and $q_0, \ldots, q_{K+6}$ of trees and a state $\tilde{a} (\in A)$ such that $q_0 \in \tau^{\tilde{a}}_{\tilde{a}}(p_0)$, $q_i \in \tau^{\tilde{a}}_{\tilde{a}}(p_i)$ $(i=1, \ldots, K+5)$, $q_{K+6} \in \tau^a_{\tilde{a}}(p_{K+6})$, $p_0 \cdot \ldots \cdot p_{K+6} = p$ and $q_0 \cdot \ldots \cdot q_{K+6} = q$.

Now there are three cases.

Firstly, we assume that there is an index j $(2 \leqq j \leqq K+6)$ for which $q_j \in T_G(Y_m)$. Then $q = q_j \cdot \ldots \cdot q_{K+6} = q_0 \cdot q_j \cdot \ldots \cdot q_{K+6} \in \tau^a_{\tilde{a}}(p_0 \cdot p_j \cdot \ldots \cdot p_{K+6})$. By the induction hypothesis concerning the tree $p_0 \cdot p_j \cdot \ldots \cdot p_{K+6}$ we have that q_a is a supertree of q.

Secondly, we suppose that there is an index j $(2 \leqq j \leqq K+5)$ for which $q_j = \#$.

It means that $q=q_0\cdot\ldots\cdot q_{j-1}\cdot q_{j+1}\cdot\ldots\cdot q_{K+6}\in\tau^a_{\bar a}(p_0\cdot\ldots\cdot p_{j-1}\cdot p_{j+1}\cdot\ldots\cdot p_{K+6})$. Again by the induction hypothesis, we get that there exists a tree $\tilde q$ for which $q=\tilde q\cdot q_a$.

Finally, we may assume that $h^{\#}(q_j)>0$ $(j=2,\ldots,K+5)$ and $q_{K+6}\in\tilde T_G(Y_m)$. Let $\bar q=q_5\cdot\ldots\cdot q_{K+6}$. Furthermore, we have that $r=q_0\cdot q_1\cdot q_2\neq q_0\cdot q_1\cdot q_2\cdot q_3=s$. By the induction hypothesis there are trees $\tilde r$ and $\tilde s$ such that $r\cdot\bar q=\tilde r\cdot q_a$ and $s\cdot\bar q=\tilde s\cdot q_a$. We know that $h(q_a)\leqq K$ and $h(\bar q)>K$. From this we obtain that the tree $\bar q$ can be given in the form $\hat q\cdot q_a$, i.e. q_a is a supertree of $\bar q$. Since $q=q_0\cdot q_1\cdot q_2\cdot q_3\cdot q_4\cdot\bar q=q_0\cdot q_1\cdot q_2\cdot q_3\cdot q_4\cdot\hat q\cdot q_a$, thus there is a tree $\tilde q$ for which $q=\tilde q\cdot q_a$. This ends the proof of our lemma.

Definition 6. An AF-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ is called *inferior* if none of the transformations induced by its states can be cut by any trees.

Take an AF-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$. Assume that the transformations induced by the states $a_1,\ldots,a_l$ can be cut and the tree q_{a_i} cuts τ^{a_i} maximally $(i=1,\ldots,l)$. For a state a, if τ^a can not be cut $(a\notin\{a_1,\ldots,a_l\})$ then let $q_a=\#$. It means that for all $a\in A$, $\bar a\in A'$, $p\in\operatorname{dom}\tau^a_{\bar a}$ and $q\in\tau^a_{\bar a}(p)$, the equality $q=\tilde q\cdot q_a$ holds under a suitable $\tilde q$.

The following lemma is valid under these notations.

Lemma 7. There is an inferior F-transducer $\bar{\mathbf{A}}$ which is equivalent to $\mathbf{A}$.

Proof. We shall show that one can construct an F-transducer

$$\bar{\mathbf{A}}=(T_F(X_n), A, T_G(Y_m), A', \bar\Sigma)$$

such that for all states $a\in A$ and $\bar a\in A'$ the following conditions are satisfied.

(1) $\operatorname{dom}\tau^a_{\mathbf{A}}=\operatorname{dom}\tau^a_{\bar{\mathbf{A}}}$ and $\operatorname{dom}\tau^a_{\mathbf{A},\bar a}=\operatorname{dom}\tau^a_{\bar{\mathbf{A}},\bar a}$.

(2) $\operatorname{dom}\tau_{\mathbf{A},a}=\operatorname{dom}\tau_{\bar{\mathbf{A}},a}$.

(3) $\{(p, q\cdot q_a)\mid q\in\tau^a_{\bar{\mathbf{A}},\bar a}(p), p\in\operatorname{dom}\tau^a_{\mathbf{A},\bar a}\}=\tau^a_{\mathbf{A},\bar a}$.

(4) $\{(p, q_a\cdot q)\mid q\in\tau_{\mathbf{A},a}(p), p\in\operatorname{dom}\tau_{\mathbf{A},a}\}=\tau_{\bar{\mathbf{A}},a}$.

From this Lemma 7 will follow. Indeed, from (3) we get that $\bar{\mathbf{A}}$ is equivalent to $\mathbf{A}$. If $\operatorname{range}\tau^a_{\bar{\mathbf{A}}}$ is a singleton then $\operatorname{range}\tau^a_{\mathbf{A}}$ is a singleton by (3), too. Using condition (i) of Definition 1 we have that for each $\bar p\in\operatorname{dom}\tau_{\mathbf{A},a}\setminus\{\#\}$, $\tau_{\mathbf{A},a}(\bar p)\subseteqq T_G(Y_m)$. Therefore, by (4), $\tau_{\bar{\mathbf{A}},a}(\bar p)\subseteqq T_G(Y_m)$. It means that (i) of Definition 1 holds for $\bar{\mathbf{A}}$. Similarly, we obtain that $\bar{\mathbf{A}}$ satisfies condition (ii). Consequently, $\bar{\mathbf{A}}$ is an AF-transducer. It is also clear that $\bar{\mathbf{A}}$ is an inferior F-transducer, too. In the opposite case we would arrive at a contradiction by assuming the maximality of the trees q_a $(a\in A)$.

Next we define the rules of $\bar{\mathbf{A}}$ in the following way.

(i) $x\to ar\in\bar\Sigma$ $(x\in X_n\cup F_0)$ if and only if
$x\to a\bar r\in\bar\Sigma$, where $r=\bar r\cdot q_a$.

(ii) $f(a_1,\ldots,a_k)\to ar\in\bar\Sigma$ $(f\in F_k, k>0)$ if and only if
$f(a_1,\ldots,a_k)\to a\bar r$, where the tree $\bar r=r(q_{a_1}(z_1),\ldots,q_{a_k}(z_k))$ is equal to $\bar r\cdot q_a$.

First, we show that the rules of $\bar\Sigma$ can be constructed. It is obvious, that this construction can be performed if the rule satisfies the assumption (i) or (ii) provided the equality $q_a=\#$.

Then let $f(a_1, ..., a_k) \to ar \in \Sigma$ $(f \in F_k,\ k>0)$ be an arbitrary rule such that $q_a \in \tilde{T}_G(Y_m) \setminus \{\#\}$. We have that for every final states $\bar{a}$ and all trees $p_i \in \operatorname{dom} \tau_{\bar{a}}^{a_i}$, $q_i \in \tau_{\bar{a}}^{a_i}(p_i)$ $(i=1, ..., k)$ the following conditions hold.

(a) $r(q_1, ..., q_k) \in \tau_{\bar{a}}^{a}(f(p_1, ..., p_k))$ and

(b) $r(q_1, ..., q_k) = r(\tilde{q}_1 \cdot q_{a_1}, ..., \tilde{q}_k \cdot q_{a_k}) =$

$= r(q_{a_1}(z_1), ..., q_{a_k}(z_k))(\tilde{q}_1, ..., \tilde{q}_k) = \bar{r}(\tilde{q}_1, ..., \tilde{q}_k)$.

Let $f(p_1, ..., p_k)=p$ and $r(q_1, ..., q_k)=q$. By Definition 4, $q=\tilde{q} \cdot q_a$. Therefore, $\bar{r}(\tilde{q}_1, ..., \tilde{q}_k)=\tilde{q} \cdot q_a$.

Let s be a tree for which there exist trees $r_1, ..., r_m \in T_G(Y_m \cup Z_k)$ and $t_1, ..., t_m \in T_G(Y_m \cup \{\#\})$ $(m>0)$ such that $s\langle r_1, ..., r_m\rangle = \bar{r}$ and $s\langle t_1, ..., t_m\rangle = q_a$, moreover, for each index j $(1 \leqq j \leqq m)$ at least one of the conditions $r_j \in Z_k$ and $t_j = \#$ holds. It means that for an arbitrary index j $(1 \leqq j \leqq m)$, $r_j(\tilde{q}_1, ..., \tilde{q}_k) = \tilde{q} \cdot t_j$.

Assume that $r_j \in Z_k$, i.e. there is an index l $(1 \leqq l \leqq k)$ satisfying $r_j = z_l$. Thus for each tree $p_l \in \operatorname{dom} \tau_{\bar{a}}^{a_l}$ and $q_l \in \tau_{\bar{a}}^{a_l}(p_l)$ the equalities $\tilde{q}_l \cdot q_{a_l} = q_l$ and $\tilde{q}_l = \tilde{q} \cdot t_j$ hold, that is t_j is a supertree of $\tilde{q}_l$.

If $t_j \in T_G(Y_m)$ then $q_l = \tilde{q} \cdot t_j \cdot q_{a_l} = t_j \cdot q_{a_l}$ implies that range τ^{a_l} is a singleton. On the other hand the symbol z_l is contained in the frontier of the tree $\bar{r}$. Therefore, it should occur in the frontier of r, too. This means that range $\tau_{a_l} \nsubseteq T_G(Y_m)$, thus by the condition (i) of Definition 1 range τ^{a_l} is not a singleton which is a contradiction. Then we have that $t_j \in \tilde{T}_G(Y_m)$.

If $t_j \neq \#$ then, by $q_l = \tilde{q} \cdot t_j \cdot q_{a_l}$, the transformation τ^{a_l} can be cut by the tree $t_j \cdot q_{a_l}$ which contradicts the maximality of q_{a_l}.

Now we have that for each index j $(1 \leqq j \leqq m)$, $t_j = \#$. It implies that $s = q_a$. Therefore, $\bar{r} = q_a\langle r_1, ..., r_m\rangle$. Using (b) we obtain that

$$\bar{r}(\tilde{q}_1, ..., \tilde{q}_k) = q_a\langle r_1(\tilde{q}_1, ..., \tilde{q}_k), ..., r_m(\tilde{q}_1, ..., \tilde{q}_k)\rangle = \tilde{q} \cdot q_a,$$

consequently, $\tilde{q} = r_j(\tilde{q}_1, ..., \tilde{q}_k)$ $(j=1, ..., m)$.

We shall prove that the trees $r_1, ..., r_m$ are equal to each other. Let $s_1, s_2 \in \{r_1, ..., r_m\}$ be arbitrary. Then the equality $s_1(\tilde{q}_1, ..., \tilde{q}_k) = s_2(\tilde{q}_1, ..., \tilde{q}_k)$ holds for each $p_i \in \operatorname{dom} \tau^{a_i}$ and $q_i \in \tau^{a_i}(p_i)$ $(i=1, ..., k)$. Let j $(1 \leqq j \leqq k)$ be an arbitrary index. Let $p_i \in \operatorname{dom} \tau^{a_i}$ and $t_i \in \tau^{a_i}(p_i)$ $(i=1, ..., k;\ i \neq j)$ be arbitrary fixed trees, moreover $\tilde{t}_j = \#$. Denote the trees $s_1(\tilde{t}_1, ..., \tilde{t}_k)$ and $s_2(\tilde{t}_1, ..., \tilde{t}_k)$ by u_j and v_j, respectively. We have that for each $p_j \in \operatorname{dom} \tau^{a_j}$ and $q_j \in \tau^{a_j}(p_j)$ the equality $\tilde{q}_j \cdot u_j = \tilde{q}_j \cdot v_j$ holds. It is obvious that $u_j \in T_G(Y_m)$ if and only if $v_j \in T_G(Y_m)$, moreover, if $u_j \in \tilde{T}_G(Y_m)$ then range τ^{a_j} is not a singleton. From this we obtain that $u_j = v_j$. It means that for all indices j $(1 \leqq j \leqq k)$ the equality $u_j = v_j$ holds, which implies that $s_1 = s_2$.

We now have that $r_1 = r_2 = ... = r_m$, and this tree is denoted by $\bar{r}$. It follows that $\bar{r} = \bar{r} \cdot q_a$, thus the rules of $\bar{\Sigma}$ can be constructed.

Consider the F-transducer $\bar{\mathbf{A}} = (T_F(X_n), A, T_G(Y_m), A', \bar{\Sigma})$ constructed in this way. We will show that $\bar{\mathbf{A}}$ has the properties (1)—(4). By the construction, it is easy to see that (1) and (2) hold. The property (3) shall be proved by induction.

Let $a\in A$ and $\bar{a}\in A'$ be arbitrary states and $p\in\operatorname{dom}\tau^a_{\mathbf{A},\bar{a}}$. Assume that $h(p)=0$. If $p\in X_n\cup F_0$ then $p\to a\tilde{q}\in\bar{\Sigma}$ if and only if $p\to a\tilde{q}\cdot q_a\in\Sigma$. Therefore, $(p,\tilde{q})\in\tau^a_{\mathbf{A},\bar{a}}$ if and only if $(p,\tilde{q}\cdot q_a)\in\tau^a_{\mathbf{A},\bar{a}}$.

If $p=\#$ then $a=\bar{a}$ and $q_a=\#$, thus $\tau^a_{\mathbf{A},\bar{a}}(p)=\tau^a_{\bar{\mathbf{A}},\bar{a}}(p)=\#$.

Assume that $p=f(p_1,\ldots,p_k)$ and $q\in\tau^a_{\mathbf{A},\bar{a}}(p)$. There is a rule $f(a_1,\ldots,a_k)\to ar\in\Sigma$ and there exist trees $q_i\in\tau^{a_i}_{\mathbf{A},\bar{a}}(p_i)$ $(i=1,\ldots,k)$ such that $q=r(q_1,\ldots,q_k)$. By the induction hypothesis we have that there are trees $\tilde{q}_i\in\tau^{a_i}_{\bar{\mathbf{A}},\bar{a}}(p_i)$ for which $\tilde{q}_i\cdot q_{a_i}=q_i$ $(i=1,\ldots,k)$. Therefore, $q=r(q_{a_1}(z_1),\ldots,q_{a_k}(z_k))(\tilde{q}_1,\ldots,\tilde{q}_k)$.

By our construction there is a rule $f(a_1,\ldots,a_k)\to a\bar{r}\in\bar{\Sigma}$, where

$$r(q_{a_1}(z_1),\ldots,q_{a_k}(z_k))=\bar{r}\cdot q_a.$$

Then $\tilde{q}=\bar{r}(\tilde{q}_1,\ldots,\tilde{q}_k)\in\tau^a_{\bar{\mathbf{A}},\bar{a}}(p)$ and $q=\tilde{q}\cdot q_a$. Similarly, we get that if $\tilde{q}\in\tau^a_{\bar{\mathbf{A}},\bar{a}}(p)$ then $\tilde{q}\cdot q_a\in\tau^a_{\mathbf{A},\bar{a}}(p)$. It means that $\overline{\mathbf{A}}$ has property (3).

Let $\bar{p}\in\operatorname{dom}\tau_{\mathbf{A},a}$ and $r\in\tau_{\mathbf{A},a}(\bar{p})$ be arbitrary trees. By the proof of (3), there is a tree $\tilde{r}\in\tau_{\bar{\mathbf{A}},a}(\bar{p})$ such that for each $p\in\operatorname{dom}\tau^a_{\mathbf{A}}$ and $q\in\tau^a_{\mathbf{A}}(p)$, $q\cdot r=\tilde{q}\cdot\tilde{r}$ and $\tilde{q}\cdot q_a=q$ under a suitable tree $\tilde{q}$. It is easy to show that $r\in T_G(Y_m)$ if and only if $\tilde{r}\in T_G(Y_m)$. It follows that if $r\in T_G(Y_m)$ then $\tilde{r}=r=q_a\cdot r$. If $r\in\tilde{T}_G(Y_m)$ then none of range $\tau^a_{\mathbf{A}}$ and range $\tau^a_{\bar{\mathbf{A}}}$ is a singleton. Using this we obtain that $\tilde{r}=q_a\cdot r$. It means that if $(\bar{p},r)\in\tau_{\mathbf{A},a}$ then $(\bar{p},q_a\cdot r)\in\tau_{\bar{\mathbf{A}},a}$. The inverse claim can be shown in a similar way.

This ends the proof of Lemma 7.

Let $\mathbf{A}=(T_F(X_n),A,T_G(Y_m),A',\Sigma_A)$ and $\mathbf{B}=(T_F(X_n),B,T_G(Y_m),B',\Sigma_B)$ be AF-transducers for which $\operatorname{dom}\tau_{\mathbf{A}}=\operatorname{dom}\tau_{\mathbf{B}}$. We construct the F-transducers $\mathbf{A}^1=(T_F(X_n),A\times C,T_G(Y_m),A'\times C',\Sigma^1_A)$ and $B^1=(T_F(X_n),B\times C,T_G(Y_m),B'\times C',\Sigma^1_B)$, where $C=A\times B$, $C'=A'\times B'$ and the sets of rules satisfy the following conditions.

(a) For each $c=(a,b)\in C$ and $x\in X_n\cup F_0$,
$x\to(a,c)q\in\Sigma^1_A$ and $x\to(b,c)r\in\Sigma^1_B$ if and only if
$x\to aq\in\Sigma_A$ and $x\to br\in\Sigma_B$.

(b) For each $f\in F_k$ $(k>0)$ and $c_i=(a_i,b_i)$ $(i=0,1,\ldots,k)$,
$f((a_1,c_1),\ldots,(a_k,c_k))\to(a_0,c_0)q\in\Sigma^1_A$ and
$f((b_1,c_1),\ldots,(b_k,c_k))\to(b_0,c_0)r\in\Sigma^1_B$ if and only if
$f(a_1,\ldots,a_k)\to a_0q\in\Sigma_A$ and $f(b_1,\ldots,b_k)\to b_0r\in\Sigma_B$.

Using a standard construction we get two connected F-transducers $\mathbf{A}^2=(T_F(X_n),\overline{A\times C},T_G(Y_m),\overline{A'\times C'},\Sigma^2_A)$ and $\mathbf{B}^2=(T_F(X_n),\overline{B\times C},T_G(Y_m),\overline{B'\times C'},\Sigma^2_B)$ such that $\mathbf{A}^2$ is equivalent to $\mathbf{A}^1$ and $\mathbf{B}^2$ is equivalent to $\mathbf{B}^1$. Moreover, using the constructions of the proofs of Lemmas 2 and 7 we obtain two inferior F-transducers $\overline{\mathbf{A}}=(T_F(X_n),\overline{A\times C},T_G(Y_m),\overline{A'\times C'},\bar{\Sigma}_A)$ and $\overline{\mathbf{B}}=(T_F(X_n),\overline{B\times C},T_G(Y_m),\overline{B'\times C'},\bar{\Sigma}_B)$ which are equivalent to $\mathbf{A}^2$ and $\mathbf{B}^2$, resp. Let us denote the inferior F-transducers $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ by $\mathbf{A}(\mathbf{B})$ and $\mathbf{B}(\mathbf{A})$, respectively. Since $\tau_{\mathbf{A}}=\tau_{\mathbf{A}(\mathbf{B})}$ both $\tau_{\mathbf{A}}$ and $\tau_{\mathbf{A}(\mathbf{B})}$ will be denoted by φ. Similarly, ψ will denote $\tau_{\mathbf{B}}$ and $\tau_{\mathbf{B}(\mathbf{A})}$.

In the next lemmas and Theorem 11 we shall use the above notations.

Lemma 8. Let $(a,b)=c$, $(\bar{a},\bar{b})=\bar{c}\in C$. Then the following conditions are satisfied:

(i) $(a, c)\in\overline{A\times C}$ if and only if $(b, c)\in\overline{B\times C}$,
(ii) $(a, c)\in\overline{A'\times C'}$ if and only if $(b, c)\in\overline{B'\times C'}$,
(iii) dom $\varphi^{a,c}=$dom $\psi^{b,c}$,
(iv) *dom* $\varphi_{a,c}=$*dom* $\psi_{b,c}$,
(v) dom $\varphi^{a,c}_{\bar a,\bar c}=$dom $\psi^{b,c}_{b,\bar c}$.

Proof. By the definitions of Σ^1_A and Σ^1_B there is a natural bijective mapping of Σ^1_A onto Σ^1_B. It is easy to see that the restriction of the above mapping to $\bar\Sigma_A$ is a bijective mapping, too. Using this the statement this lemma is obvious.

In Lemmas 9 and 10 and in Theorem 11 we assume that the AF-transducers **A** and **B** are equivalent i.e., $\varphi=\psi$. Then dom $\tau_{\mathbf{A}}=$dom $\tau_{\mathbf{B}}$, thus we may use the above notations and Lemma 8.

Lemma 9. Let $c=(a, b)\in C$ and $\bar p\in dom\,\varphi_{a,c}$ be arbitrary. Assume that the AF-transducer **A** is functional. Then $\varphi_{a,c}(\bar p)\in\tilde T_G(Y_m)$ if and only if $\psi_{b,c}(\bar p)\subseteq\subseteq T_G(Y_m)$.

Proof. First of all we note that, by Lemma 3, the transformations ψ, $\psi^{b,c}$ and $\psi_{b,c}$ are mappings. Assume that there is a tree $\bar p$ for which the conclusion of this lemma does not hold. Let $\varphi_{a,c}(\bar p)=q$ and $\psi_{b,c}(\bar p)=r$. Then exactly one of q and r is in $T_G(Y_m)$, say $r\in T_G(Y_m)$ and $q\in\tilde T_G(Y_m)$. We have that $\bar p\neq\#$. Thus by condition (i) of Definition 1, range $\varphi^{a,c}$ is not a singleton. It means that there are trees $p_1, p_2(\in$dom $\varphi^{a,c})$ for which $q_1=\varphi^{a,c}(p_1)\neq\varphi^{a,c}(p_2)=q_2$. Then $q_i\cdot q=$ $=\varphi(p_i\cdot\bar p)=\psi(p_i\cdot\bar p)=r$ $(i=1, 2)$, consequently, $q_1\cdot q=q_2\cdot q$, which contradicts the assumption $q_1\neq q_2$. Similarly, we arrive at a contradiction by assuming $r\in\tilde T_G(Y_m)$ and $q\in T_G(Y_m)$.

Lemma 10. If **A** is functional, then $\varphi^{a,c}=\psi^{b,c}$ for all $(a, b)=c(\in C)$.

Proof. First we note that if (a, c) and (b, c) are final states then the equality $\varphi=\psi$ implies $\varphi^{a,c}=\psi^{b,c}$. We may assume that (a, c) and (b, c) are not final states. By Lemma 9, range $\varphi^{a,c}$ is a singleton if and only if range $\psi^{b,c}$ is a singleton, too. If both range $\varphi^{a,c}$ and range $\psi^{b,c}$ are singletons then the equality $\varphi^{a,c}(p)=y_1=\psi^{b,c}(p)$ holds for each tree $p\in$dom $\varphi^{a,c}$. Therefore, in this case $\varphi^{a,c}=\psi^{b,c}$.

Suppose that range $\varphi^{a,c}$ is not a singleton. By the note following Definition 1. we have that *range* $\varphi_{a,c}\not\subseteq T_G(Y_m)$ i.e., there is a tree $\bar p\in dom\,\varphi_{a,c}$ satisfying the inclusion $\varphi_{a,c}(\bar p)\in\tilde T_G(Y_m)$. Let $\varphi_{a,c}(\bar p)=\bar q$ and $\psi_{b,c}(\bar p)=\bar r$. In the same way as in the proof of Lemma 7, one can see that there exist trees $s\in\tilde T_G(Y_m)$, $r_1, \ldots, r_m$ and $q_1, \ldots, q_m$ $(m>0)$ such that the equalities $\bar r=s\langle r_1, \ldots, r_m\rangle$ and $\bar q=s\langle q_1, \ldots, q_m\rangle$ hold, moreover, at least one of q_i and r_i is $\#$ for each index i $(1\leqq i\leqq m)$. It is easy to show that $q_i, r_i\in\tilde T_G(Y_m)$ $(i=1, \ldots, m)$.

Next we prove that all the r_i and q_i are equal to $\#$ $(i=1, \ldots, m)$. Let i be an arbitrary index $(1\leqq i\leqq m)$. Assume that $q_i=\#$ and $r_i\in\tilde T_G(Y_m)\setminus\{\#\}$. For each final state $(\bar a, \bar c)$ $((\bar a, \bar b)=\bar c)$ and for all trees $p\in$dom $\varphi^{a,c}_{\bar a,\bar c}(p)=$dom $\psi^{b,c}_{b,\bar c}$, if $q\in\varphi^{a,c}_{\bar a,\bar c}(p)$ and $r\in\psi^{b,c}_{b,\bar c}(p)$ then

$$\varphi_{\bar a,\bar c}(p\cdot\bar p)=q\cdot\bar q=s\langle q\cdot q_1, \ldots, q\cdot q_m\rangle \quad\text{and}$$
$$\psi_{b,\bar c}(p\cdot\bar p)=r\cdot\bar r=s\langle r\cdot r_1, \ldots, r\cdot r_m\rangle.$$

Since $(\bar{a}, \bar{c})$ and $(\bar{b}, \bar{c})$ are final states $\varphi^{\bar{a},\bar{c}}=\psi^{\bar{b},\bar{c}}$, which implies $\varphi_{\bar{a},\bar{c}}(p\cdot\bar{p})=\psi_{\bar{b},\bar{c}}(p\cdot\bar{p})$. Therefore, $r\cdot r_i=q\cdot q_i$, i.e. $r\cdot r_i=q$. It means that the transformation $\varphi^{a,c}$ can be cut by the tree r_i, which is a contradiction. Similarly, the assumptions $r_i=\#$ and $q_i\in\tilde{T}_G(Y_m)\setminus\{\#\}$ imply the equality $q_i=\#$.

Now we have that $\bar{r}=\bar{q}=s$ and $r=q$. It means that for each

$$p\in\operatorname{dom}\varphi^{a,c}(\subseteq\operatorname{dom}\varphi^{a,c}_{\bar{a},\bar{c}}),\ \varphi^{a,c}(p)=\psi^{b,c}(p)$$

holds. This ends the proof of Lemma 10.

Theorem 11. If the AF-transducer **A** is functional then the inferior F-transducers **A**(**B**) and **B**(**A**) are isomorphic.

Proof. Let us define a mapping $\mu\colon \overline{A\times C}\to\overline{B\times C}$, such that for an arbitrary state $(a, c)(\in\overline{A\times C})$ the equality $\mu(a, c)=(b, c)$ holds if $c=(a, b)$. It is clear that μ is a bijective mapping of $\overline{A\times C}$ onto $\overline{B\times C}$, moreover, $\mu(\overline{A'\times C'})=\overline{B'\times C'}$.

Next suppose that $x\to(a, c)q\in\bar{\Sigma}_A$ $(x\in X_n\cup F_0)$, where $c=(a, b)$. We have $x\in\operatorname{dom}\varphi^{a,c}$ thus $x\in\operatorname{dom}\psi^{b,c}$. By Lemma 10, $q=\varphi^{a,c}(x)=\psi^{b,c}(x)$ implies $x\to(b, c)q\in\bar{\Sigma}_B$. Similarly, if $x\to(b, c)r\in\bar{\Sigma}_B$ then we get $x\to(a, c)r\in\bar{\Sigma}_A$.

Let $f((a_1, c_1), \ldots, (a_k, c_k))\to(a_0, c_0)q\in\bar{\Sigma}_A$ where $c_i=(a_i, b_i)$ $(i=0, 1, \ldots, k)$. By the construction of **A**(**B**) and **B**(**A**) we know that there is a rule of the form $f((b_1, c_1), \ldots, (b_k, c_k))\to(b_0, c_0)r$ in $\bar{\Sigma}_B$. Let $p_i(\in\operatorname{dom}\varphi^{a_i,c_i}=\operatorname{dom}\psi^{b_i,c_i})$ be arbitrary trees $(i=1, \ldots, k)$ and let j be an arbitrary index $(1\leqq j\leqq k)$. We define the trees s_i $(i=1, \ldots, k)$ in the following way. If $i=j$ then $s_i=\#$, otherwise $s_i=\varphi^{a_i,c_i}(p_i)\left(=\psi^{a_i,c_i}(p_i)\right)$ $(i=1, \ldots, k)$.

Denote by $\bar{q}_j$ and $\bar{r}_j$ the tree $q(s_1, \ldots, s_k)$ and $r(s_1, \ldots, s_k)$, respectively. We have that $\varphi^{a_j,c_j}(p_j)=\psi^{b_j,c_j}(p_j)$ for each $p_j\in\operatorname{dom}\varphi^{a_j,c_j}$. From this it follows easily that the equality $\bar{r}_j=\bar{q}_j$ holds. Since j is arbitrary we get $r=q$. It means that $f((b_1, c_1), \ldots, (b_k, c_k))\to(b_0, c_0)q\in\bar{\Sigma}_B$.

Similarly, one can see that if $f((b_1, c_1), \ldots, (b_k, c_k))\to(b_0, c_0)r\in\bar{\Sigma}_B$ then the rule $f((a_1, c_1), \ldots, (a_k, c_k))\to(a_0, c_0)r$ is in $\bar{\Sigma}_A$.

Therefore, the inferior F-transducers **A**(**B**) and **B**(**A**) are isomorphic.

The next corollary is known from [2], where the result has been achieved in a different way.

Corollary 12. There exists an algorithm to decide for an arbitrary F-transducer $\overline{\mathbf{B}}$ and a functional F-transducer $\overline{\mathbf{A}}$ whether they are equivalent, i.e. $\tau_{\overline{\mathbf{A}}}=\tau_{\overline{\mathbf{B}}}$.

Proof. Let **A** and **B** be AF-transducers equivalent to $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$, respectively. Clearly, $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ are equivalent if and only if so are **A** and **B**. By Theorem 11, $\tau_{\mathbf{A}}=\tau_{\mathbf{B}}$ if and only if $\operatorname{dom}\tau_{\mathbf{A}}=\operatorname{dom}\tau_{\mathbf{B}}$ and the inferior transducers **A**(**B**) and **B**(**A**) are isomorphic. It is known that the equality $\operatorname{dom}\tau_{\mathbf{A}}=\operatorname{dom}\tau_{\mathbf{B}}$ is decidable (c.f. [3, 4]). Obviously, **A**(**B**) and **B**(**A**) can be constructed. Moreover the isomorphism of these inferior transducers can be verified. Thus the statement of Corollary 12 is valid.

Acknowledgement. The author wishes to thank Professor F. Gécseg for his valuable suggestions.

References

[1] ELGELFRIET, J., Bottom-up and top-down tree transformations — A comparison, Math. Systems Theory, v. 9, 1975, pp. 198—231.
[2] ÉSIK, Z., Decidability results concerning tree transducers. I, Acta Cybernet., v. 5, 1980, pp. 1—20.
[3] GÉCSEG, F.—STEINBY, M., A faautomaták algebrai elmélete I., Matematikai Lapok, v. 26, 1975, pp. 169—207.
[4] GÉCSEG, F.—STEINBY, M., A faautomaták algebrai elmélete II., Matematikai Lapok, v. 27, 1976—1979, pp. 283—336.

Received Apr. 19, 1984

TATABÁNYA COAL MINES
VÉRTANÚK TERE 1
2800 TATABÁNYA, HUNGARY

On the equivalence of the frontier-to-root tree transducers II.

By Z. ZACHAR

In this paper we continue our study started in [6] about the equivalent and isomorphic frontier-to-root transducers (*F*-transducers). First we introduce the superior *F*-transducer which can be seen the dual of the inferior *F*-transducer from part I. Then we deal with a subclass of the class of deterministic *F*-transducers, namely the class of normalized *F*-transducers. It will be proved that the strongly normalized forms of equivalent deterministic *F*-transducers are isomorphic.

Since this paper connects with [6] closely thus we use the notions, notations and results of part I.

1. Notions and notations

Take an arbitrary positive integer k. Let $p_1, p_2 \in T_F(X_n \cup Z_k)$ be arbitrary trees and $z_i \in Z_k$. Then the *z_i-product* $p_1 \cdot_i p_2$ of p_1 by p_2 is the tree

$$p_2(z_1, \ldots, z_{i-1}, p_1, z_{i+1}, \ldots, z_k).$$

For an *F*-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ and sets $A_i \subseteq A$ $(i=0, \ldots, k)$ we denote by $\tau_{\mathbf{A},A_1,\ldots,A_k}^{A_0}$ the transformation induced by

$$(T_F(X_n \cup Z_k), A, T_G(Y_m \cup Z_k), A_0, \Sigma \cup \{z_i \to a_i z_i | a_i \in A_i,\ i=1, \ldots, k\}).$$

Finally, when we will refer to a definition or a result from a part of our paper if the serial number of the part is I then it will be marked otherwise it will not be.

2. Superior *F*-transducers

Definition 1. Let $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ be an AF-transducer. The transformation induced by the state $a(\in A)$ can *be increased* by the tree $q^a \in \tilde{T}_G(Y_m) \setminus \{\#\}$ if for all $p \in dom\ \tau_{\mathbf{A},a}$ and $q \in \tau_{\mathbf{A},a}(p)$, there is a tree $\tilde{q} \in T_G(Y_m \cup \{\#\})$ satisfying $q=q^a \cdot \tilde{q}$, provided that range $\tau_{\mathbf{A}}^a$ is not a singleton. The tree q^a increases the transformation $\tau_{\mathbf{A}}^a$ maximally if the tree q^a is a proper subtree of a tree $\bar{q}^a$ then $\tau_a^{\mathbf{A}}$ cannot be increased by $\bar{q}^a$.

Definition 2. An AF-transducer $A=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ is called a *superior* AF-transducer if none of the transformations induced by its states can be increased by any trees.

Take an AF-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$. Assume that for each state $a\in A$ the tree q^a increases τ^a maximally if τ^a can be increased and $q^a=\sharp$ otherwise. It means that for all $a\in A$, $p\in dom\ \tau_a$ and $q\in\tau_a(p)$ there is a tree $\tilde{q}$ such that $q=q^a\cdot\tilde{q}$. We suppose that the tree q^a is given for every state $a(\in A)$. Then the following lemma is valid under these notations.

Lemma 3. There is a superior AF-transducer $\overline{\mathbf{A}}=(T_F(X_n), A, T_G(Y_m), A', \bar{\Sigma})$ which is equivalent to $\mathbf{A}$.

Proof. We shall show that one can construct an AF-transducer

$$\overline{\mathbf{A}} = (T_F(X_n), A, T_G(Y_m), A', \bar{\Sigma})$$

such that for each state $a\in A$ the following conditions hold.

(1) $\mathrm{dom}\ \tau^a_{\overline{\mathbf{A}}}=\mathrm{dom}\ \tau^a_{\mathbf{A}}$.

(2) $\mathrm{dom}\ \tau_{\mathbf{A},a}=\mathrm{dom}\ \tau_{\overline{\mathbf{A}},a}$ and $dom\ \tau_{\mathbf{A},a}=dom\ \tau_{\overline{\mathbf{A}},a}$.

(3) $\{(p, q\cdot q^a)|q\in\tau^a_{\mathbf{A}}(p),\ p\in\mathrm{dom}\ \tau^a_{\mathbf{A}}\}=\tau^a_{\overline{\mathbf{A}}}$.

(4) $\{(p, q^a\cdot q)|q\in\tau_{\overline{\mathbf{A}},a}(p),\ p\in\mathrm{dom}\ \tau_{\mathbf{A},a}\}=\tau_{\mathbf{A},a}$.

In a way similar to that in the proof of Lemma I.7 we can see that $\overline{\mathbf{A}}$ is an equivalent superior AF-transducer for $\mathbf{A}$.

Next we define the rules of $\bar{\Sigma}$ in the following way:

(i) $x\to ar\in\Sigma$ $(x\in X_n\cup F_0)$ if and only if $x\to a\bar{r}\in\bar{\Sigma}$ where $\bar{r}=r\cdot q^a$,

(ii) $f(a_1, \ldots, a_k)\to ar\in\Sigma$ $(f\in F_k,\ k>0)$ if and only if $f(a_1, \ldots, a_k)\to a\bar{r}\in\bar{\Sigma}$ where the tree $\bar{r}(q^{a_1}(z_1), \ldots, q^{a_k}(z_k))$ equals the tree $r\cdot q^a$.

It is clear that this construction can be made for each rule of form (i). Assume that $f(a_1, \ldots, a_k)\to ar\in\Sigma$ $(f\in F_k,\ k>0)$.

Then let $p^j\in\mathrm{dom}\ \tau^{a_j}_{\mathbf{A}}$ and $t_j\in\tau^{a_j}_{\mathbf{A}}(p^j)$ be arbitrary fixed trees $(j=1, \ldots, k)$. For each index j $(1\leqq j\leqq k)$ we use the following notations:

$$p_j = f(p^1, \ldots, p^{j-1}, \sharp, p^{j+1}, \ldots, p^k),$$

$$r_j = r(t_1, \ldots, t_{j-1}, \sharp, t_{j+1}, \ldots, t_k) \quad \text{and}$$

$$\bar{r}_j = r_j\cdot q^a.$$

It is sufficient to show that for each index j $(1\leqq j\leqq k)$ there is a tree $\bar{q}_j$ such that $\bar{r}_j=q^{a_j}\cdot\bar{q}_j$. From this we obtain easily that the tree $\bar{r}$ with

$$r\cdot q^a = \bar{r}(q^{a_1}(z_1), \ldots, q^{a_k}(z_k))$$

can be constructed.

Let j be an arbitrary index $(1\leqq j\leqq k)$. If $\bar{r}_j\in T_G(Y_m)$ or $q^{a_j}=\sharp$ then let $\bar{q}_j=\bar{r}_j$. In this case our statement holds obviously.

We may assume that $\bar{r}_j\in\tilde{T}_G(Y_m)$ and $\bar{q}^{a_j}\in\tilde{T}_G(Y_m)\setminus\{\sharp\}$. If range τ^a is a singleton then by the construction from Lemma I.2 the tree r is in $T_G(Y_m)$. It follows that $\bar{r}_j\in T_G(Y_m)$ which is a contradiction. It means that range τ^a is not a

singleton. From this we obtain that there are trees $p \in dom\, \tau_a$ and $q \in \tau_a(p)$ such that $q \in \tilde{T}_G(Y_m)$. It implies that $r_j \cdot q \in \tau_{a_j}(p_j \cdot p)$ and $r_j \cdot q \in \tilde{T}_G(Y_m)$. By Definition 2 we know that $r_j \cdot q = q^{a_j} \cdot \tilde{q}$ under a suitable $\tilde{q}$. It means that $\tilde{q} \in \tilde{T}_G(Y_m)$. Moreover, one of the inclusions $q^{a_j} \in \mathrm{sub}\,(r_j)$ and $r_j \in sub\,(q^{a_j})$ holds.

Firstly, assume that $q^{a_j} \in \mathrm{sub}\,(r_j)$. Then there exists a tree $\bar{q} \in \tilde{T}_G(Y_m)$ for which $r_j = q^{a_j} \cdot \bar{q}$. In this case let $\bar{q}_j = \bar{q} \cdot q^a$. It means that $\bar{r}_j = r_j \cdot q^a = q^{a_j} \cdot \bar{q} \cdot q^a = q^{a_j} \cdot \bar{q}_j$.

Secondly, assume that $r_j \in sub\,(q^{a_j})$. Then there is a tree $\bar{q} \in \tilde{T}_G(Y_m) \setminus \{\sharp\}$ for which $q^{a_j} = r_j \cdot \bar{q}$. We have that for each tree $p \in dom\, \tau_a$ and $q \in \tau_a(p)$, the inclusion $r_j \cdot q \in \tau_{a_j}(p_j \cdot p)$ holds. Moreover, there is a tree $\tilde{q}$ such that $r_j \cdot q = q^{a_j} \cdot \tilde{q}$. From this we obtain that $r_j \cdot q = r_j \cdot \bar{q} \cdot \tilde{q}$. Since $r_j \in \tilde{T}_G(Y_m)$ the equality $q = \bar{q} \cdot \tilde{q}$ holds, too. It means that τ^a can be increased by the tree $\bar{q}$.

On the other hand we have that $q = q^a \cdot \tilde{\tilde{q}}$ under a suitable tree $\tilde{\tilde{q}}$. Since the tree q^a increases τ^a maximally thus from the two equalities above we get $\bar{q} \in \mathrm{sub}\,(q^a)$ i.e., there exists a $\bar{\bar{q}}$ for which $\bar{q} \cdot \bar{\bar{q}} = q^a$. Let $\bar{q}_j = \bar{\bar{q}}$. It follows that $\bar{r}_j = r_j \cdot q^a = r_j \cdot \bar{q} \cdot \bar{\bar{q}} = q^{a_j} \cdot \bar{\bar{q}} = q^{a_j} \cdot \bar{q}_j$.

It means that our statement is valid, thus the rules of $\bar{\Sigma}$ can be constructed.

Finally, one can see easily that the F-transducer $\bar{\mathbf{A}} = (T_F(X_n), A, T_G(Y_m), A', \bar{\Sigma})$ constructed in this way satisfies conditions (1)—(4).

This ends the proof of Lemma 3.

Lemma 4. There is an algorithm to decide for each AF-transducer

$$\mathbf{A} = (T_F(X_n), A, T_G(Y_m), A', \Sigma)$$

and arbitrary state $a (\in A)$ whether the transformation τ^a can be increased. Moreover, every tree q^a can be given effectively which increases τ^a.

Proof. We have that if the transformation τ^a can be increased by the tree q^a then $h(q^a) \leqq \min\,(\tau_a(p) | p \in dom\, \tau_a)$. It means that the number of trees which increases τ^a is finite. Moreover, by the proof of Lemma 3 it is easy to see that for each tree q^a the transformation τ^a is increased by q^a if and only if the rules of Σ can be rewritten according to the conditions (i)—(ii) from the proof of Lemma 3. From this the statement of our lemma is obtained obviously.

3. Normalized F-transducers

Definition 5. A deterministic AF-transducer $\mathbf{A} = (T_F(X_n), A, T_G(Y_m), A', \Sigma)$ is called a *normalized* F-transducer (NF-transducer) if conditions (i) and (ii) below hold.

(i) For every state $a \in A$, range τ^a is either a singleton or infinite.

(ii) For all states a, $\bar{a}$ if both range τ^a and range $\tau^{\bar{a}}$ are infinite, $dom\, \tau_a = dom\, \tau_{\bar{a}}$ and there exist trees $q, \bar{q} \in T_G(Y_m)$ such that for each tree $\bar{p} \in dom\, \tau_a$, $q \cdot \tau_a(\bar{p}) = \bar{q} \cdot \tau_{\bar{a}}(\bar{p})$ then at least one of the following conditions are satisfied.

(ii$_1$) There are trees $r, \bar{r} \in \tilde{T}_G(Y_m)$ such that at least one of them is equal to the tree $\sharp$ and for each tree $\bar{p} \in dom\, \tau_a$ the equality $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$ holds.

(ii$_2$) The sets range $\tau^a \cap q$ and range $\tau^{\bar{a}} \cap \bar{q}$ are empty.

The next lemma, in a different form, can be found in [2]. The proof can be performed easily thus it is omitted.

Lemma 6. Let $q_j, r_j \in T_G(Y_m \cup Z)$ be arbitrary trees $(j=1, \ldots, 5)$. For each positive integer i the equalities (1)—(7) imply the equality (8).

(1) $r_1 \cdot_i r_5 = q_1 \cdot_i q_5$

(2) $r_1 \cdot_i r_2 \cdot_i r_5 = q_1 \cdot_i q_2 \cdot_i q_5$

(3) $r_1 \cdot_i r_3 \cdot_i r_5 = q_1 \cdot_i q_3 \cdot_i q_5$

(4) $r_1 \cdot_i r_4 \cdot_i r_5 = q_1 \cdot_i q_4 \cdot_i q_5$

(5) $r_1 \cdot_i r_2 \cdot_i r_3 \cdot_i r_5 = q_1 \cdot_i q_2 \cdot_i q_3 \cdot_i q_5$

(6) $r_1 \cdot_i r_2 \cdot_i r_4 \cdot_i r_5 = q_1 \cdot_i q_2 \cdot_i q_4 \cdot_i q_5$

(7) $r_1 \cdot_i r_3 \cdot_i r_4 \cdot_i r_5 = q_1 \cdot_i q_3 \cdot_i q_4 \cdot_i q_5$

(8) $r_1 \cdot_i r_2 \cdot_i r_3 \cdot_i r_4 \cdot_i r_5 = q_1 \cdot_i q_2 \cdot_i q_3 \cdot_i q_4 \cdot_i q_5$

Lemma 7. For any deterministic F-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ an equivalent NF-transducer can be constructed.

Proof. Let $K=\max(h(\tau^a(p))|p\in\mathrm{dom}\,\tau^a,\ a\in A,\ h(p)\leqq\|A\|)$ and
$L_1=\max(h(\tau_a(p))|p\in dom\,\tau_a,\ a\in A,\ h(p)\leqq 4\cdot\|A\|^2)$,
$L_2=\max(h(\tau_a(p))|p\in dom\,\tau_a,\ a\in A,\ h(p)\leqq 2\cdot\|A\|,\ h^{\#}(p)\leqq\|A\|)$ and
$L=L_1+L_2$.
Moreover, set $Q=\{q|q\in T_G(Y_m), h(q)\leqq\max(K, L)\}$ and $C=Q\cup\{\#\}$.
Construct the deterministic F-transducer

$$\mathbf{A}^1=(T_F(X_n), A\times C, T_G(Y_m), A'\times C, \Sigma^1)$$

such that $x\to(a, c)r\in\Sigma^1$ if and only if $x\to ar\in\Sigma$ and $c=r$, moreover,

$$f((a_1, c_1), \ldots, (a_k, c_k))\to(a, c)r\in\Sigma^1$$

if and only if $f(a_1, \ldots, a_k)\to a\bar{r}\in\Sigma$ where c and r are defined in the following way. Let $q=\bar{r}(c_1(z_1), \ldots, c_k(z_k))$. If $q\in Q$ then $c=q$ otherwise $c=\#$. If $a\notin A'$ and $q\in Q$ then $r=y_1$ otherwise $r=q$. It is obvious that $\mathbf{A}$ and $\mathbf{A}^1$ are equivalent. Eliminating surplus states and rules in a standard way we get a connected deterministic F-transducer $\mathbf{B}=(T_F(X_n), B, T_G(Y_m), B', \Sigma_B)$ where $B\subseteqq A\times C$, $B'\subseteqq A'\times C$ and $\Sigma_B\subseteqq\Sigma^1$. It is clear that $\mathbf{B}$ and $\mathbf{A}^1$ are equivalent.

We will show that $\mathbf{B}$ is an NF-transducer. Take an arbitrary state $b=(a, c)\in B$. By our construction it is clear that $\mathrm{dom}\,\tau_{\mathbf{A},a}=\mathrm{dom}\,\tau_{\mathbf{B},b}$ and if $p\in\mathrm{dom}\,\tau^b_{\mathbf{B}}$ then $p\in\mathrm{dom}\,\tau^a_{\mathbf{A}}$, moreover, if $c=\#$ then the equality $\tau^a_{\mathbf{A}}(p)=\tau^b_{\mathbf{B}}(p)$ holds, too.

Assume that $c=\#$. Then for each tree $p\in\mathrm{dom}\,\tau^b_{\mathbf{B}}(\subseteqq\mathrm{dom}\,\tau^a_{\mathbf{A}})$ the inequality $h(\tau^b_{\mathbf{B}}(p))>\max(K, L)\geqq K$ holds. It follows that there are trees p_1, p_2, p_3 and a state $\tilde{a}$ such that $p=p_1\cdot p_2\cdot p_3$, $p_1\in\mathrm{dom}\,\tau^{\tilde{a}}_{\mathbf{A}}$, $p_2\in dom\,\tau^{\tilde{a}}_{\mathbf{A},\tilde{a}}$, $p_3\in dom\,\tau^a_{\mathbf{A},\tilde{a}}$ and $h^{\#}(\tau^{\tilde{a}}_{\mathbf{A},\tilde{a}}(p_2))>0$, $\tau^a_{\mathbf{A},\tilde{a}}(p_3)\in\tilde{T}_G(Y_m)$. From this we obtain that $p^k=p_1\cdot p_2^k\cdot p_3\in\mathrm{dom}\,\tau^a_{\mathbf{A}}$ $(k=1, 2, \ldots)$ and the trees $\tau^a_{\mathbf{A}}(p^k)$ are pairwise different. Since $\mathrm{range}\,\tau^b_{\mathbf{B}}=\mathrm{range}\,\tau^a_{\mathbf{A}}\setminus Q$ thus $\mathrm{range}\,\tau^b_{\mathbf{B}}$ is infinite. Moreover, we have that for all trees $\bar{p}\in\mathrm{dom}\,\tau_{\mathbf{B},b}$ and $p\in\mathrm{dom}\,\tau^b_{\mathbf{B}}$ the equality $\tau^a_{\mathbf{A}}(p)\cdot\tau_{\mathbf{A},a}(\bar{p})=\tau^b_{\mathbf{B}}(p)\cdot\tau_{\mathbf{B},b}(\bar{p})$ holds. From this we obtain that $\tau_{\mathbf{A},a}=\tau_{\mathbf{B},b}$.

Furthermore, we know that if $c \in Q$ then range $\tau_{\mathbf{B}}^{b}$ is a singleton and for each tree $\bar{p} \in \operatorname{dom} \tau_{\mathbf{B},b} \setminus \{\sharp\}$, $\tau_{\mathbf{B},b}(\bar{p}) \in T_G(Y_m)$. It follows that **B** is a deterministic AF-transducer and condition (i) of Definition 5 holds for **B**.

Then we have to prove that condition (ii) of Definition 5 can be satisfied. Take arbitrary states $b, \bar{b} \in B$ and trees $q, \bar{q} \in T_G(Y_m)$. Let $b=(a, c)$ and $\bar{b}=(\bar{a}, \bar{c})$. Assume that $dom\, \tau_b = dom\, \tau_{\bar{b}}$, both range τ^b and range $\tau^{\bar{b}}$ are infinite, moreover, for each tree $\bar{p} \in dom\, \tau_b$ the equality $q \cdot \tau_b(\bar{p}) = \bar{q} \cdot \tau_{\bar{b}}(\bar{p})$ holds. In this case we have that $c = \bar{c} = \sharp$ and $dom\, \tau_a = dom\, \tau_{\bar{a}}$.

It is sufficient to show that if at least one of two trees q and $\bar{q}$ is higher than L then the following condition $(\mathrm{ii}_1)'$ holds.

$(\mathrm{ii}_1)'$ There are trees $r, \bar{r} \in \tilde{T}_G(Y_m)$ such that at least one of them is equal to the tree $\sharp$ and for each tree $\bar{p} \in dom\, \tau_a \cap R$ the equality $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$ holds where $R = \{\bar{p} \mid \bar{p} \in \tilde{T}_F(X_n), h(\bar{p}) \leqq 4 \cdot \|A\|^2\}$.

Now we prove this statement. First we show that $h(q), h(\bar{q}) > L_1$. Assume that $h(q) > L = L_1 + L_2$. It is clear that there is a tree $\bar{p} \in dom\, \tau_a$ for which $h(\bar{p}) \leqq 2 \cdot \|A\|$, $h^{\sharp}(\bar{p}) \leqq \|A\|$ and $\tau_a(\bar{p}) \in \tilde{T}_G(Y_m)$. Since $h(\tau_{\bar{a}}(\bar{p})) \leqq L_2$ and $q \cdot \tau_a(\bar{p}) = \bar{q} \cdot \tau_{\bar{a}}(\bar{p})$ thus $h(q \cdot \tau_a(\bar{p})) > L$ and $h(\bar{q} \cdot \tau_{\bar{a}}(\bar{p})) \leqq h(\bar{q}) + L_2$. It follows that $h(\bar{q}) > L_1$. Similarly, the inequality $h(\bar{q}) > L$ implies $h(q) > L_1$.

We have that there is a tree $\bar{p}$ for which $h(\bar{p}) \leqq 4 \cdot \|A\|^2$ and at least one of the trees $\tau_a(\bar{p})$ and $\tau_{\bar{a}}(\bar{p})$ is in $\tilde{T}_G(Y_m)$. In this case there exist an $s \in \tilde{T}_G(Y_m)$ and $q_1, \ldots, q_m, \bar{q}_1, \ldots, \bar{q}_m \in T_G(Y_m \cup \{\sharp\})$ $(m \geqq 1)$ such that the equalities $\tau_a(\bar{p}) = s\langle q_1, \ldots, q_m \rangle$ and $\tau_{\bar{a}}(\bar{p}) = s\langle \bar{q}_1, \ldots, \bar{q}_m \rangle$ hold, moreover, for each index j $(1 \leqq j \leqq m)$ at least one of the trees q_j and $\bar{q}_j$ equals $\sharp$. It means that $q \cdot q_j = \bar{q} \cdot \bar{q}_j$ $(j = 1, \ldots, m)$. Since $h(q_j), h(\bar{q}_j) \leqq L_1$ we get $q_j, \bar{q}_j \in \tilde{T}_G(Y_m)$ $(j = 1, \ldots, m)$.

Let j be an arbitrarily fixed index $(1 \leqq j \leqq m)$ and let $r = q_j$ and $\bar{r} = \bar{q}_j$. It follows that $q \cdot r = \bar{q} \cdot \bar{r}$. We show that for each tree $\bar{p} \in dom\, \tau_a \cap R$ the equality $r \cdot \tau_a(p) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$ holds.

Take an arbitrary tree $\bar{p} \in dom\, \tau_a \cap R$. If both $\tau_a(\bar{p})$ and $\tau_{\bar{a}}(\bar{p})$ are in $T_G(Y_m)$ then $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$ because of the equality $q \cdot \tau_a(\bar{p}) = \bar{q} \cdot \tau_{\bar{a}}(\bar{p})$.

In the opposite case the equalities $\tau_a(\bar{p}) = s\langle q_1, \ldots, q_m \rangle$ and $\tau_{\bar{a}}(\bar{p}) = s\langle \bar{q}_1, \ldots, \bar{q}_m \rangle$ hold where the trees s, $q_1, \ldots, q_m, \bar{q}_1, \ldots, \bar{q}_m$ satisfy the above conditions. Similarly, we have that $q \cdot q_j = \bar{q} \cdot \bar{q}_j$ and $q_j, \bar{q}_j \in \tilde{T}_G(Y_m)$ $(j = 1, \ldots, m)$.

Assume that $r = \sharp$, consequently, $q = \bar{q} \cdot \bar{r}$. It follows that $\bar{q} \cdot \bar{r} \cdot q_j = \bar{q} \cdot \bar{q}_j$ $(j = 1, \ldots, m)$. If $\bar{r} = \sharp$ then $\bar{q} \cdot q_j = \bar{q} \cdot \bar{q}_j$. Since $h(\bar{q}) > L_1$ and $h(q_j), h(\bar{q}_j) \leqq L_1$ thus $q_j = \bar{q}_j = \sharp$ $(j = 1, \ldots, m)$. From this we obtain $\tau_a(\bar{p}) = \tau_{\bar{a}}(\bar{p})$ i.e. $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$. We may suppose that $\bar{r} \neq \sharp$. Then $\bar{q}_j \neq \sharp$, because in the opposite case $\bar{q} \cdot \bar{r} \cdot q_j = \bar{q}$ which is a contradiction $(j = 1, \ldots, m)$. It means that for each index j $(1 \leqq j \leqq m)$, $q_j = \sharp$ and $\bar{q} \cdot \bar{r} = \bar{q} \cdot \bar{q}_j$. From this we obtain that $\bar{r} = \bar{q}_j$ $(j = 1, \ldots, m)$. It implies that $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$.

From the assumption $\bar{r} = \sharp$ we arrive at the equality $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$ in a similar way.

Now we can prove that conditions (ii) of Definition 5 are satisfied. If $h(q), h(\bar{q}) \leqq L$ then (ii_2) holds because each tree of both range τ^b and range $\tau^{\bar{b}}$ is higher than L. In the opposite case condition $(\mathrm{ii}_1)'$ holds. We will show by induction that $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$ for each tree $\bar{p} \in dom\, \tau_a$.

If $h(\bar{p}) \leqq 4 \cdot \|A\|^2$ then $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$ by $(\mathrm{ii}_1)'$. Now let $h(\bar{p}) > 4 \cdot \|A\|^2$. We have that there are trees p_1, p_2, p_3, p_4, p_5 and states $\tilde{a}, \bar{\tilde{a}} \in A$ such that

$\bar{p}=p_1 \cdot^2 p_2 \cdot^2 p_3 \cdot^2 p_4 \cdot^2 p_5$ and $p_1 \in \operatorname{dom} \tau_a^{\bar{a}} \cap \operatorname{dom} \tau_{\bar{a}}^{\tilde{\bar{a}}}$, $p_i \in \operatorname{dom} \tau_{a,\bar{a}}^{\bar{a}} \cap \operatorname{dom} \tau_{\bar{a},\tilde{\bar{a}}}^{\tilde{\bar{a}}}$ $(i=2, 3, 4)$, $p_5 \in \operatorname{dom} \tau_{a,\bar{a}} \cap \operatorname{dom} \tau_{\bar{a},\tilde{a}}$, where $p_i \in T_F(X_n \cup Z_2)$ $(i=1, \ldots, 5)$ and the symbol z_2 occurs exactly once in the frontier of the tree p_i $(i=2, \ldots, 5)$.
Let

$$q_1 = r \cdot \tau_a^{\bar{a}}(p_1), \quad r_1 = \bar{r} \cdot \tau_{\bar{a}}^{\tilde{\bar{a}}}(p_1),$$

$$q_2 = \tau_{a,\bar{a}}^{\bar{a}}(p_2), \quad r_2 = \tau_{\bar{a},\tilde{a}}^{\tilde{\bar{a}}}(p_2),$$

$$q_3 = \tau_{a,\bar{a}}^{\bar{a}}(p_3), \quad r_3 = \tau_{\bar{a},\tilde{a}}^{\tilde{\bar{a}}}(p_3),$$

$$q_4 = \tau_{a,\bar{a}}^{\bar{a}}(p_4), \quad r_4 = \tau_{\bar{a},\tilde{a}}^{\tilde{\bar{a}}}(p_4),$$

$$q_5 = \tau_{a,\bar{a}}(p_5), \quad r_5 = \tau_{\bar{a},\tilde{a}}(p_5).$$

By the induction hypothesis it is clear that the trees r_i, q_i $(i=1, \ldots, 5)$ satisfy the conditions of Lemma 6. It means that $q_1 \cdot^2 q_2 \cdot^2 q_3 \cdot^2 q_4 \cdot^2 q_5 = r_1 \cdot^2 r_2 \cdot^2 r_3 \cdot^2 r_4 \cdot^2 r_5$ i.e., $r \cdot \tau_a(\bar{p}) = \bar{r} \cdot \tau_{\bar{a}}(\bar{p})$.

We have that $\tau_a = \tau_b$ and $\tau_{\bar{a}} = \tau_{\bar{b}}$. It follows that for each tree $\bar{p} \in dom\, \tau_b$ the equality $r \cdot \tau_b(\bar{p}) = \bar{r} \cdot \tau_{\bar{b}}(\bar{p})$ holds. It means that **B** is an *NF*-transducer. Consequently the statement of this lemma is valid.

Lemma 8. Let $\overline{\mathbf{A}} = (T_F(X_n), A, T_G(Y_m), A', \bar{\Sigma}_A)$ be a deterministic *F*-transducer. Then there is an superior *NF*-transducer **B** which is equivalent to $\overline{\mathbf{A}}$.

Proof. By Lemma 3 we can construct a superior *F*-transducer

$$\mathbf{A} = (T_F(X_n), A, T_G(Y_m), A', \Sigma_A)$$

with $\tau_{\mathbf{A}} = \tau_{\bar{\mathbf{A}}}$. From the proof of Lemma 3 one can see that **A** is deterministic, too. Next we consider the *NF*-transducer $\mathbf{B} = (T_F(X_n), B, T_G(Y_m), B', \Sigma_B)$ constructed for **A** by Lemma 7. From the proof we have that for each state $b \in B$ if range $\tau_{\mathbf{B}}^b$ is not a singleton then there exists a state $a \in A$ such that $\tau_{\mathbf{A},a} = \tau_{\mathbf{B},b}$. It follows that $\tau_{\mathbf{B}}^b$ can not be increased by any tree q^b, because in the opposite case the transformation $\tau_{\mathbf{A}}^a$ is increased by q^b which is a contradiction. It means that **B** is a superior *NF*-transducer equivalent to $\overline{\mathbf{A}}$.

Definition 9. Let $\mathbf{A} = (T_F(X_n), A, T_G(Y_m), A', \Sigma)$ be a superior *NF*-transducer. We say that the state $\bar{a} (\in A)$ can *be substituted* by the state $a (\in A)$ if the condition (i) holds or there is a tree $\bar{q} \in T_G(Y_m)$ such that the conditions (ii_1)—(ii_6) are satisfied.

(i) $\tau_a = \tau_{\bar{a}}$, and if range τ^a is a singleton then range τ^a = range $\tau^{\bar{a}}$.
(ii_1) $dom\, \tau_a = dom\, \tau_{\bar{a}}$.
(ii_2) range τ^a is infinite.
(ii_3) range $\tau^{\bar{a}}$ is a singleton.
(ii_4) For each tree $\bar{p} \in dom\, \tau_a \setminus \{\sharp\}$ the equality $\bar{q} \cdot \tau_a(\bar{p}) = \tau_{\bar{a}}(\bar{p})$ holds.
(ii_5) If $\bar{a} \in A'$ then range $\tau^{\bar{a}} = \bar{q}$.
(ii_6) If there is a state $\bar{\bar{a}} (\in A \setminus \{a, \bar{a}\})$ for which $dom\, \tau_a = dom\, \tau_{\bar{\bar{a}}}$ and range $\tau^{\bar{\bar{a}}}$ is infinite, moreover, there exist trees $q, \bar{q} \in T_G(Y_m)$ such that for each tree $\bar{p} \in dom\, \tau_a$ the equality $q \cdot \tau_a(\bar{p}) = \bar{q} \cdot \tau_{\bar{\bar{a}}}(\bar{p})$ holds then either $q \neq \bar{q}$ or $\tau_a(\bar{p}) = \tau_{\bar{\bar{a}}}(\bar{p})$ for each tree $\bar{p} \in dom\, \tau_a$.

We note that $\tau_a=\tau_{\bar{a}}$ if and only if $dom\,\tau_a=dom\,\tau_{\bar{a}}$ and for each tree $\bar{p}\in dom\,\tau_a$ the equality $\tau_a(\bar{p})=\tau_{\bar{a}}(\bar{p})$ holds.

Definition 10. A superior NF-transducer $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma)$ is called a *strongly normalized* F-transducer (SNF-transducer) if none of the states can be substituted by another state.

Theorem 11. For each deterministic F-transducer

$$\overline{\mathbf{A}} = (T_F(X_n), \bar{A}, T_G(Y_m), \bar{A}', \bar{\Sigma}_A)$$

an equivalent SNF-transducer can be constructed.

Proof. By Lemma 8 we construct a superior NF-transducer

$$\mathbf{A} = (T_F(X_n), A, T_G(Y_m), A', \Sigma_A)$$

which is equivalent to $\overline{\mathbf{A}}$. Next we will show that by rewriting rules and eliminating states an equivalent SNF-transducer is obtained.

Assume that the states $a, \bar{a}\in A$ satisfy the condition (i) of Definition 9. Then we construct the F-transducer $\mathbf{A}^1=(T_F(X_n), A\setminus\{\bar{a}\}, T_G(Y_m), A'\setminus\{\bar{a}\}, \Sigma_A^1)$ in the following way. Let us eliminate the rules of Σ_A wherein the state $\bar{a}$ is in the left side. Then we replace the state $\bar{a}$ by a in each rule. It is clear that $\mathbf{A}^1$ is deterministic. Moreover, for each state $\tilde{a}\in A\setminus\{a, \bar{a}\}$, $\tau_{A,\tilde{a}}=\tau_{A^1,\tilde{a}}$ and $\tau^{\tilde{a}}_{\mathbf{A}}=\tau^{\tilde{a}}_{\mathbf{A}^1}$ hold. We have that $\mathrm{dom}\,\tau^{a}_{\mathbf{A}^1}=\mathrm{dom}\,\tau^{a}_{\mathbf{A}}\cup\mathrm{dom}\,\tau^{\bar{a}}_{\mathbf{A}}$ and $\mathrm{range}\,\tau^{a}_{\mathbf{A}^1}=\mathrm{range}\,\tau^{a}_{\mathbf{A}}\cup\mathrm{range}\,\tau^{\bar{a}}_{\mathbf{A}}$. From this one can easily show that $\mathbf{A}^1$ is a superior NF-transducer equivalent to $\mathbf{A}$. It means that for $\mathbf{A}$ we construct an equivalent superior NF-transducer $\mathbf{B}=(T_F(X_n), B, T_G(Y_m), B', \Sigma_B)$ such that there are no states $b, \bar{b}\in B$ satisfying condition (i) of Definition 9.

Next we assume that there are states $a, \bar{a}\in B(\subseteq A)$ and a tree $\bar{q}\in T_G(Y_m)$ for which the conditions (ii_1)—(ii_6) hold. In this case we eliminate the rules containing the state $\bar{a}$ in their left side. Then we replace by $a\bar{q}$ the right side of rules wherein the state $\bar{a}$ occurs. Let $\mathbf{B}^1=(T_F(X_n), B\setminus\{\bar{a}\}, T_G(Y_m), B'\setminus\{\bar{a}\}, \Sigma_B^1)$ be the F-transducer obtained this way. By the construction it is obvious that $\mathbf{B}^1$ is deterministic. We have that for each state $\tilde{a}\in B\setminus\{a, \bar{a}\}$, $\tau_{\mathbf{B}^1,\tilde{a}}=\tau_{\mathbf{B},\tilde{a}}$ and $\tau^{\tilde{a}}_{\mathbf{B}^1}=\tau^{\tilde{a}}_{\mathbf{B}}$ hold. Moreover, $\mathrm{dom}\,\tau^{a}_{\mathbf{B}^1}=\mathrm{dom}\,\tau^{a}_{\mathbf{B}}\cup\mathrm{dom}\,\tau^{\bar{a}}_{\mathbf{B}}$ and $\mathrm{range}\,\tau^{a}_{\mathbf{B}^1}=\mathrm{range}\,\tau^{a}_{\mathbf{B}}\cup\bar{q}$. It is clear that $\mathbf{B}^1$ is a superior AF-transducer equivalent to $\mathbf{B}$.

We will show that $\mathbf{B}^1$ is normalized. By the construction of $\mathbf{B}^1$ condition (i) of Definition 5 holds. Let $b, \bar{\bar{b}}\in B\setminus\{\bar{a}\}$ be arbitrary states of $\mathbf{B}^1$. Assume that $dom\,\tau_{\mathbf{B}^1,b}=dom\,\tau_{\mathbf{B}^1,\bar{\bar{b}}}$, both $\mathrm{range}\,\tau^{b}_{\mathbf{B}^1}$ and $\mathrm{range}\,\tau^{\bar{\bar{b}}}_{\mathbf{B}^1}$ are infinite, moreover, there are trees $q, \bar{q}\in T_G(Y_m)$ such that for each tree $\bar{p}\in dom\,\tau_{\mathbf{B}^1,b}$, $q\cdot\tau_{\mathbf{B}^1,b}(\bar{p})=\bar{q}\cdot\tau_{\mathbf{B}^1,\bar{\bar{b}}}(\bar{p})$. If none of the states b, $\bar{\bar{b}}$ is a then by the above connections it is obvious that condition (ii) holds.

We may assume that $b=a$. In this case we know that for $\mathbf{B}$ condition (ii_6) are satisfied by the states a, $\bar{a}$, $\bar{\bar{b}}$ and the trees q, $\bar{q}$, $\bar{\bar{q}}$. Furthermore, we have that the equality $\tau_{\mathbf{B},a}(\bar{p})=\tau_{\mathbf{B},\bar{\bar{b}}}(\bar{p})$ does not hold for each tree $\bar{p}\in dom\,\tau_{\mathbf{B},a}$. Indeed, in the opposite case $\tau_{\mathbf{B},a}=\tau_{\mathbf{B},\bar{\bar{b}}}$ which is a contradiction. By condition (ii_6) it means that $q\neq\bar{q}$. From this we obtain that $\mathrm{range}\,\tau_{\mathbf{B}^1,a}\cap q=\emptyset$ and $\mathrm{range}\,\tau_{\mathbf{B}^1,\bar{\bar{b}}}\cap\bar{\bar{q}}=\emptyset$. Consequently, condition (ii) of Definition 5 holds for $\mathbf{B}^1$ thus $\mathbf{B}^1$ is a superior normalized F-transducer.

Applying this construction we can get an *SNF*-transducer which is equivalent to $\overline{\mathbf{A}}$.

Let $\mathbf{A}=(T_F(X_n), A, T_G(Y_m), A', \Sigma_A)$ and $\mathbf{B}=(T_F(X_n), B, T_G(Y_m), B', \Sigma_B)$ be *SNF*-transducers such that $\mathbf{A}$ and $\mathbf{B}$ are equivalent. First we construct the inferior *AF*-transducers $\mathbf{A}(\mathbf{B})$ and $\mathbf{B}(\mathbf{A})$ as in part I. Next we consider the superior *AF*-transducers $\overline{\mathbf{A}}(\mathbf{B})=(T_F(X_n), \overline{A\times C}, T_G(Y_m), \overline{A'\times C'}, \bar{\Sigma}_A)$ and

$$\overline{\mathbf{B}}(\mathbf{A}) = (T_F(X_n), \overline{B\times C}, T_G(Y_m), \overline{B'\times C'}, \bar{\Sigma}_B)$$

which are constructed from $\mathbf{A}(\mathbf{B})$ and $\mathbf{B}(\mathbf{A})$ by Lemma 3, respectively. We have that $\overline{A\times C}\subseteq A\times C$, $\overline{A'\times C'}\subseteq A'\times C'$ and $\overline{B\times C}\subseteq B\times C$, $\overline{B'\times C'}\subseteq B'\times C'$ where $C=A\times B$ and $C'=A'\times B'$. From Theorem I.11 we know that $\mathbf{A}(\mathbf{B})$ and $\mathbf{B}(\mathbf{A})$ are isomorphic. By conditions (i)—(iv) from the proof of Lemma 3, it follows that $\overline{\mathbf{A}}(\mathbf{B})$ and $\overline{\mathbf{B}}(\mathbf{A})$ are isomorphic, too. It means that $\tau_{\mathbf{A}}=\tau_{\overline{\mathbf{A}}(\mathbf{B})}$ thus both of these transformations shall be denoted by φ. Similarly, ψ can be used instead of $\tau_{\mathbf{B}}$ and $\tau_{\overline{\mathbf{B}}(\mathbf{A})}$.

The next lemmas are valid under the above notations.

Lemma 12. For each state $(a, c)\in\overline{A\times C}$ $(c=(a, b))$ the following conditions hold.

(i) $dom\ \varphi_a=dom\ \varphi_{a,c}=dom\ \psi_{b,c}=dom\ \psi_b$.

(ii) If range $\varphi^{a,c}$ is infinite then for each tree $\bar{p}\in dom\ \varphi_a$ the equalities $\varphi_a(\bar{p})=\varphi_{a,c}(\bar{p})=\psi_{b,c}(\bar{p})=\psi_b(\bar{p})$ hold.

(iii) If range $\varphi^{a,c}$ is a singleton then there are trees $q, \bar{q}\in T_G(Y_m)$ such that for each tree $\bar{p}\in dom\ \varphi_a\setminus\{\#\}$ the equalities $q\cdot\varphi_a(\bar{p})=\varphi_{a,c}(\bar{p})=\psi_{b,c}(\bar{p})=\bar{q}\cdot\psi_{b,c}(\bar{p})$ hold, moreover $q=\varphi^a(p)$ and $\bar{q}=\psi^b(p)$ where $p\in\mathrm{dom}\ \varphi^{a,c}$.

Proof. Let $(a, c)\in\overline{A\times C}$ be an arbitrary state where $c=(a, b)$. Let $p\in\mathrm{dom}\ \varphi^{a,c}$ be a fixed tree. Then $p\in\mathrm{dom}\ \varphi^a\cap\mathrm{dom}\ \psi^{b,c}\cap\mathrm{dom}\ \psi^b$. Let $\bar{p}\in dom\ \varphi_a$. Since $p\cdot\bar{p}\in\mathrm{dom}\ \varphi$ the tree $\bar{p}$ is in $dom\ \varphi_{a,c}$. Consequently, $dom\ \varphi_a\subseteq dom\ \varphi_{a,c}$. In the same way one can prove the inclusions $dom\ \varphi_a\subseteq dom\ \varphi_{a,c}\subseteq dom\ \psi_{b,c}\subseteq dom\ \psi_b\subseteq dom\ \varphi_a$. From this we obtain that condition (i) holds.

Assume that range $\varphi^{a,c}$ is infinite. It follows that range $\psi^{b,c}$ is infinite, too. Then there are trees $p_1, p_2\in\mathrm{dom}\ \varphi^{a,c}$ such that $\varphi^{a,c}(p_1)\neq\varphi^{a,c}(p_2)$. For each tree $\bar{p}\in dom\ \varphi_a$ the equality $\varphi^{a,c}(p_i)\cdot\varphi_{a,c}(\bar{p})=\psi^{b,c}(p_i)\cdot\psi_{b,c}(\bar{p})$ holds $(i=1, 2)$. In a similar way as in the proof of Lemma 3, we can obtain that there exist trees $r, \bar{r}$ such that at least one of them equals the tree $\#$ and for each tree $\bar{p}\in dom\ \varphi_a$, $r\cdot\varphi_a(\bar{p})=\bar{r}\cdot\varphi_{a,c}(\bar{p})$.

On the other hand we have that both $\overline{\mathbf{A}}(\mathbf{B})$ and $\overline{\mathbf{B}}(\mathbf{A})$ are superior *F*-transducers. It means that $r=\bar{r}=\#$ i.e., for each tree $\bar{p}\in dom\ \varphi_a$ the equality $\varphi_a(\bar{p})=\varphi_{a,c}(\bar{p})$ holds.

Furthermore we know that range $\psi^{b,c}$ is infinite. In the same way we get that for each tree $\bar{p}\in dom\ \psi_b$, $\psi_{b,c}(\bar{p})=\psi_b(\bar{p})$. Since $\overline{\mathbf{A}}(\mathbf{B})$ and $\overline{\mathbf{B}}(\mathbf{A})$ are isomorphic it follows that condition (ii) of our lemma holds.

Next we assume that range $\varphi^{a,c}$ is a singleton. Let $p\in\mathrm{dom}\ \varphi^{a,c}$ be an arbitrarily fixed tree. We have that $p\in\mathrm{dom}\ \varphi^a$. Let $q=\varphi^a(p)$. It means that for each tree $\bar{p}\in dom\ \varphi_a\setminus\{\#\}$ the equalities $q\cdot\varphi_a(\bar{p})=\varphi(p\cdot\bar{p})=\varphi^{a,c}(p)\cdot\varphi_{a,c}(\bar{p})=\varphi_{a,c}(\bar{p})$ hold. In this case range $\psi^{b,c}$ is a singleton, too. It follows that if $\psi^{b,c}(p)=\bar{q}$ then

for each tree $\bar{p}\in dom\,\psi_a\setminus\{\sharp\}$, $\bar{q}\cdot\psi_b(\bar{p})=\psi_{b,c}(\bar{p})$. It is clear that if $a\in A'$ then b, (a, c), (b, c) are final states. From this we obtain $q=\bar{q}$. It means that condition (iii) holds, too.

Lemma 13. For each state $a\in A$ there is exactly one state $b(\in B)$ satisfying the inclusion $(a,(a,b))\in\overline{A\times C}$, and conversely, for each state $b\in B$ there is exactly one state $a(\in A)$ with $(b,(a,b))\in\overline{B\times C}$. Moreover, if $(a,c)\in\overline{A\times C}$ $(c=(a,b))$ then for each tree $\bar{p}\in dom\,\varphi_a$ the equalities $\varphi_a(\bar{p})=\varphi_{a,c}(\bar{p})=\psi_{b,c}(\bar{p})=\psi_b(\bar{p})$ hold.

Proof. Let $a\in A$ be an arbitrary state. Denote by B_a the set

$$\{b|c=(a,b),\ (a,c)\in\overline{A\times C}\}.$$

It is clear that B_a is a nonvoid set.

Firstly, we assume that range φ^a is infinite. Then there are trees $p_i\in$dom φ^a $(i=1, 2, \ldots)$ such that the trees $\varphi^a(p_i)$ are pairwise different. Moreover, we know that there exists a state $b_i(\in B_a)$ such that $p_i\in$dom ψ^{b_i} $(i=1, 2, \ldots)$. Since B_a is a finite set of states there are indices k, l $(k<l)$ satisfying $b_k=b_l$. Denote by b this state. Let $c=(a, b)$. It is clear that neither range $\varphi^{a,c}$ nor range $\psi^{b,c}$ are a singleton. By Lemma 12 we get that for each tree $\bar{p}\in dom\,\varphi_a$ the equalities $\varphi_a(\bar{p})=$ $=\varphi_{a,c}(\bar{p})=\psi_{b,c}(\bar{p})=\psi_b(\bar{p})$ hold.

Next we show that the set B_a is a singleton. Assume that there is a state $\bar{b}\in B_a$ differing from b. Let $\bar{c}=(a,\bar{b})$. Now there are three cases.

First, suppose that range $\varphi^{a,\bar{c}}$ is infinite. By Lemma 12 we have that $\varphi_a(\bar{p})=$ $=\varphi_{a,\bar{c}}(\bar{p})=\psi_{\bar{b},\bar{c}}(\bar{p})=\psi_{\bar{b}}(\bar{p})$ hold for each tree $\bar{p}\in dom\,\varphi_a$. It means that the state $\bar{b}$ can be substituted by b which is a contradiction because **B** is an *SNF*-transducer.

In the second case assume that both range $\varphi^{a,\bar{c}}$ and range $\psi^{\bar{b}}$ are singleton. Then we know that for each tree $\bar{p}\in dom\,\varphi_a\setminus\{\sharp\}$ the equalities $q\cdot\varphi_a(\bar{p})=\varphi_{a,\bar{c}}(\bar{p})=$ $=\psi_{\bar{b},\bar{c}}(\bar{p})=\psi_{\bar{b}}(\bar{p})$ hold, where $q=\varphi^a(p)$ and $p\in$dom $\varphi^{a,\bar{c}}$. It is clear that $q=$range $\psi^{\bar{b}}$ if $\bar{b}$ is a final state. From this we obtain $q\cdot\psi_b(\bar{p})=\psi_{\bar{b}}(\bar{p})$ for each tree $\bar{p}\in dom\,\psi^b\setminus\{\sharp\}$. Since the state $\bar{b}$ cannot be substituted by the state b and conditions (ii_1)—(ii_5) of Definition 9 hold for the states b, $\bar{b}$ and the tree q condition (ii_6) can not be satisfied. It means that there is a state $\bar{\bar{b}}\in B\setminus\{b,\bar{b}\}$ and a tree $\bar{q}\in T_G(Y_m)$ for which $dom\,\psi_b=dom\,\psi^{\bar{\bar{b}}}_{\bar{b}}$ and range $\psi^{\bar{\bar{b}}}$ is infinite, moreover, for each tree $\bar{p}\in dom\,\psi_b$ the equality $q\cdot\psi_b(\bar{p})=\bar{q}\cdot\psi^{\bar{\bar{b}}}_{\bar{b}}(\bar{p})$ holds. One can see easily that there is a state $\bar{\bar{a}}\in A\setminus\{a,\bar{a}\}$ such that $dom\,\varphi_{\bar{\bar{a}}}=dom\,\psi^{\bar{\bar{b}}}_{\bar{b}}$ and for each tree $\bar{p}\in dom\,\psi^{\bar{\bar{b}}}_{\bar{b}}$, $\varphi_{\bar{\bar{a}}}(\bar{p})=\psi^{\bar{\bar{b}}}_{\bar{b}}(\bar{p})$. It implies that for each tree $\bar{p}\in dom\,\varphi_a$, $q\cdot\varphi_a(\bar{p})=\bar{q}\cdot\varphi_{\bar{\bar{a}}}(\bar{p})$. Since **A** is an *NF*-transducer condition (ii) of Definition 5 has to hold. We have that $q\cap$range $\varphi^a\neq\emptyset$ thus there are trees r, $\bar{r}\in\tilde{T}_G(Y_m)$ such that $r\cdot\varphi_a(\bar{p})=\bar{r}\cdot\varphi_{\bar{\bar{a}}}(\bar{p})$ for each tree $\bar{p}\in dom\,\varphi_a$, where at least one of the trees r, $\bar{r}$ equals $\sharp$. It is clear that $r=\bar{r}=\sharp$ because **A** is a superior *NF*-transducer. It implies that for each tree $\bar{p}\in dom\,\varphi_a$ the equality $\varphi_a(\bar{p})=\varphi_{\bar{\bar{a}}}(\bar{p})$ holds which is a contradiction.

In the third case suppose that range $\varphi^{a,\bar{c}}$ is a singleton and range $\psi^{\bar{b}}$ is infinite. We have that for each tree $\bar{p}\in dom\,\varphi_a\setminus\{\sharp\}$ the equalities $q\cdot\varphi_a(\bar{p})=\varphi_{a,\bar{c}}(\bar{p})=$ $=\psi_{\bar{b},\bar{c}}(\bar{p})=\bar{q}\cdot\psi_{\bar{b}}(\bar{p})$ hold where $q\in$range φ^a and $\bar{q}\in$range $\psi^{\bar{b}}$. We have that if a is a final state then $\bar{b}$ is also a final state and $q=\bar{q}$. It implies that for each tree $\bar{p}\in dom\,\psi_b$, $q\cdot\psi_b(\bar{p})=\bar{q}\cdot\psi_{\bar{b}}(\bar{p})$. From Definition 5 we obtain that either for each tree $\bar{p}\in dom\,\psi_b$ the equality $\psi_b(\bar{p})=\psi_{\bar{b}}(\bar{p})$ holds or range $\psi^{\bar{b}}\cap\bar{q}=\emptyset$. It contra-

dicts the above statements. It means that if range φ^a is infinite then B_a is a singleton.

Similarly, we can show that for each state $b\in B$ if range ψ^b is infinite then there is exactly one state $a\in A$ satisfying the inclusion $(b, c)\in\overline{B\times C}$ where $c=(a, b)$. Moreover, for each tree $\bar{p}\in dom\,\psi_b$ the equalities $\varphi_a(\bar{p})=\varphi_{a,c}(\bar{p})=\psi_{b,c}(\bar{p})=$ $=\psi_b(\bar{p})$ hold.

Secondly, we may assume that range φ^a is a singleton. It is clear that $B_a\neq\emptyset$ and for each state $b\in B_a$ range ψ_b is a singleton, too. Let $b\in B_a$ be arbitrary. Then for each tree $\bar{p}\in dom\,\varphi_a$ the equalities $\varphi_a(\bar{p})=\varphi_{a,c}(\bar{p})=\psi_{b,c}(\bar{p})=\psi_b(\bar{p})$ hold where $c=(a, b)$. We have that b is a final state if and only if a is a final state. It implies that range $\varphi^a=$range ψ^b. From this and Definition 9 we get that B_a is a singleton.

In a similar way we can see that for each $b\in B$ if range ψ^b is a singleton then there is exactly one state $a\in A$ such that $(b, c)\in\overline{B\times C}$ $(c=(a, b))$ and for each tree $\bar{p}\in dom\,\varphi_a$ the equalities $\varphi_a(\bar{p})=\varphi_{a,c}(\bar{p})=\psi_{b,c}(\bar{p})=\psi_b(\bar{p})$ hold. This ends the proof of Lemma 13.

Lemma 14. The *SNF*-transducers **A** and **B** are isomorphic.

Proof. Let us define a mapping $\mu\colon A\to B$ such that $\mu(a)=b$ if and only if $(a, (a, b))\in\overline{A\times C}$. By Lemma 13 it is clear that μ is a bijective mapping of A onto B, moreover, $\mu(A')=B'$.

Next suppose that $x\to aq\in\Sigma_A$ $(x\in X_n\cup F_0)$ and $b=\mu(a)$. We have that $x\to br\in\Sigma_B$ and for each tree $\bar{p}\in dom\,\varphi_a=dom\,\psi_b$ the equality $\varphi_a(\bar{p})=\psi_b(\bar{p})$ holds. It implies that $q\cdot\varphi_a(\bar{p})=\varphi(x\cdot\bar{p})=\psi(x\cdot\bar{p})=r\cdot\psi_b(\bar{p})$. From this we can obtain that $q=r$. It means that $x\to bq\in\Sigma_B$. Similarly, if $x\to br\in\Sigma_B$ and $a=\mu^{-1}(b)$ then $x\to ar\in\Sigma_A$.

Let $f(a_1, ..., a_k)\to a_0q\in\Sigma_A$ where $f\in F_k$ $(k>0)$ and $a_i\in A$ $(i=0, 1, ..., k)$. We have that there is a rule of the form $f(b_1, ..., b_k)\to b_0r$ in Σ_B where $b_i=\mu(a_i)$ $(i=0, 1, ..., k)$. Moreover, it is clear that dom $\varphi^{a_i}=$dom ψ^{b_i}, and for each tree $p_i\in dom\,\varphi^{a_i}$, $\varphi^{a_i}(p_i)=\psi^{b_i}(p_i)$ $(i=0, 1, ..., k)$. From the proof of Lemma 13 we know that if range φ^{a_0} is a singleton then $q=$range $\varphi^{a_0}=$range $\psi^{b_0}=r$.

Next we may assume that range φ^{a_0} is infinite. In this case we have that there is a tree $\bar{p}\in dom\,\varphi_{a_0}$ for which $\varphi_{a_0}(\bar{p})\in\tilde{T}_G(Y_m)$. Let $p_i\in$dom φ^{a_i} $(i=1, ..., k)$ be arbitrary trees and let j be an arbitrary index $(1\leqq j\leqq k)$. We define the trees s_i, t_i $(i=1, ..., k)$ in the following way. If $i=j$ then $s_i=t_i=\sharp$, otherwise $s_i=p_i$ and $t_i=\varphi^{a_i}(p_i)=\psi^{b_i}(p_i)$ $(i=1, ..., k)$. Denote by $\bar{s}_j$, $\bar{q}_j$ and $\bar{r}_j$ the trees $f(s_1, ..., s_k)$, $q(t_1, ..., t_k)$ and $r(t_1, ..., t_k)$, respectively. By Lemma 13 we have that the equalities $\bar{q}_j\cdot\varphi_{a_0}(\bar{p})=\varphi_{a_j}(\bar{s}_j\cdot\bar{p})=\psi_{b_j}(\bar{s}_j\cdot\bar{p})=\bar{r}_j\cdot\psi_{b_0}(\bar{p})$ and $\varphi_{a_0}(\bar{p})=\psi_{b_0}(\bar{p})$ hold. It follows that $\bar{q}_j=\bar{r}_j$. Since j is arbitrary we get $r=q$. It means that $f(b_1, ..., b_k)\to b_0q\in\Sigma_B$.

Similarly, one can see that if $f(b_1, ..., b_k)\to b_0r\in\Sigma_B$ then $f(a_1, ..., a_k)\to a_0r\in\Sigma_A$ where $a_i=\mu^{-1}(b_i)$ $(i=0, 1, ..., k)$. Therefore, the *SNF*-transducers **A** and **B** are isomorphic.

By this lemma we get the following theorem.

Theorem 15. The *SNF*-transducers **A** and **B** are equivalent if and only if they are isomorphic.

References

[1] ENGELFRIET, J., Bottom-up and top-down tree transformations — A comparison, Math. Systems Theory, v. 9, 1975, pp. 198—231.
[2] ENGELFRIET, J., Some open questions and recent results on tree transducers and languages, Formal language theory, ed. by R. V. Book, Academic Press, 1980, pp. 241—286.
[3] ÉSIK, Z., Decidability results conserning tree transducers I, Acta Cybernet., v. 5, 1980, pp. 1—20.
[4] GÉCSEG, F.—STEINBY, M., A faautomaták algebrai elmélete I, Matematikai Lapok, v. 26, 1975, pp. 169—207.
[5] GÉCSEG, F.—STEINBY, M., A faautomaták algebrai elmélete II, Matematikai Lapok, v. 27, 1976—1979, pp. 283—336.
[6] ZACHAR, Z., On the equivalence of the frontier-to-root transducers I, Acta Cybernet., to appear.

Received Apr. 19, 1984

TATABÁNYA COAL MINES
VÉRTANÚK TERE 1
2800 TATABÁNYA, HUNGARY

Systems of linear equations over a bounded chain

K. PEEVA

§ 1. Introduction

The equivalence, reduction and minimization are classical problems for the theory of abstract automata. They are completely studied for deterministic, nondeterministic and stochastic automata (8). These problems are still open for fuzzy automata because there does not exist a polynomial time algorithm for solving systems of linear equations over a bounded chain.

Let $\mathbf{L}=(L, \vee, \wedge, 0, 1)$ be a bounded chain (5) with underlying linearly ordered set L, the greatest element 1 and the smallest element 0. We shall write $\mathbf{L}$ instead of $\mathbf{L}=(L, \vee, \wedge, 0, 1)$.

For a given set D we denote by $|D|$ its cardinality.

Let $\mathbf{L}$ be given. Let $I\neq\emptyset$, $J\neq\emptyset$ be sets of indices. We write $B\in L^{I\times J}$ for the matrix $B=(b_{ij})$ where $b_{ij}=b(i,j)$ is the (i,j)-th entry of a map $b\colon I\times J\to L$.

Let J be finite set and $A=(a_{ij})\in L^{I\times J}$, $B=(b_{jk})\in L^{J\times K}$ be given. The matrix $C=A\cdot B=(c_{ik})\in L^{I\times K}$ is called a *product* of A and B if

$$c_{ik}=\bigvee_{p=1}^{|J|}(a_{ip}\wedge b_{pk}) \text{ for each } i\in I,\ k\in K.$$

Obviously $c_{ik}\in\{a_{ip}\colon p\in J\}\cup\{b_{pk}\colon p\in J\}$.

Let $A=(X, Q, Y, M)$ be a fuzzy automaton (7, 9) with input alphabet X, state set Q, output alphabet Y and set of the step-behaviour matrices

$$M=\{M(x/y)\colon x\in X, y\in Y\}.$$

Each $M(x/y)=(m(x/y)_{qq'})\in L^{|Q|\times|Q|}$ and $m(x/y)_{qq'}$ is the grade of membership of a transition to state q' under input x assuming the output is y and the start state is q. If X, Q, Y are finite sets then A is finite fuzzy automaton.

For any set D we write D^* for the free monoid on D with the empty word $e\in D^*$ as the identity element. For $(u, v)\in X^*\times Y^*$ we write (u/v) if the number of the letters in u is equal to that of the letters in v.

Let $A=(X, Q, Y, M)$ be a finite automaton. The expression

$$M(u/v)=M(x_1/y_1)\ldots M(x_k/y_k),\quad u=x_1\ldots x_k\in X^*,\quad v=y_1\ldots y_k\in Y^*$$

defines the operation of A for the pair of words (u/v). Let

$$M(u/v) = (m(u/v)_{qq'}).$$

For the given automaton A let us consider its behaviour matrix

$$B^* = (b(u/v)_q), \quad q \in Q, \quad (u/v) \in X^* \times Y^*,$$

where $b(u/v)_q = \bigvee_{q' \in Q} m(u/v)_{qq'}$ is the grade of membership of the output v upon the input u when the start state is q. The matrix B^* is semi-infinite with $|Q|$ rows.

It is well-known (7, 9) that there exists a finite submatrix B of B^* with linearly independent columns. For the problems of equivalence, reduction and minimization of fuzzy automata the main question is how to compute B from B^*. That means for any column in B^* we have to answer whether it is a $\vee$—$\wedge$-linear combination of the previous columns in B^*. If we can solve systems of linear equations over a bounded chain then we can compute B from B^*.

In this paper the attention is concentrated on computing a solution of the system of linear equations over **L**. The main result is (see Algorithm 3 and the Theorem corresponding to it) that there exists a polynomial time algorithm for solving a system of linear equations over a bounded chain. An extension of this result for some semirings is given in (6).

We would like to remark that the classical methods (5) for solving systems of linear equations over a field are not useful here because **L** is not a field. Since the conjugate matrix for a given matrix in **L** does not exist in general, the ideas of (1) can not be applied. As our problem is essentially different from the extremal linear programing (10) these results can not be implemented.

Further we shall use without explicit explanation the concept of computational complexity as described in (3) and the properties of chains according to (5).

§ 2. Linear equations over L

In order to determine the general solution of the system we consider first a linear equation in **L**.

By $A \cdot X = b$ we denote the following linear equation

$$(a_1 \wedge x_1) \vee \ldots \vee (a_n \wedge x_n) = b \tag{1}$$

with coefficients $A = (a_j) \in L^{\{1\} \times J}$, unknowns $X = (x_j) \in L^{J \times \{1\}}$ and a constant $b \in L$. Here $\{1\}$ stands for the singleton set and we assume $|J| = n \in \mathbf{N}$.

The matrix $X^0 = (x_j^0) \in L^{J \times \{1\}}$ is a *point solution* of (1) if and only if $A \cdot X^0 = b$ holds. If there exists X^0 with $A \cdot X^0 = b$ then the equation (1) is called *solvable*, otherwise it is *unsolvable*. An n-tuple $(X_1, \ldots, X_n)$ of intervals $X_i \subseteq L$ is called an *interval solution* of (1) if every n-tuple $(x_1, \ldots, x_n)$ with $x_i \in X_i$ is a point solution of (1) and $(X_1, \ldots, X_n)$ is maximal with respect to this property.

Let the equation (1) be given and

$$S = \{j \in J: \ a_j < b\}, \quad E = \{j \in J: \ a_j = b\}, \quad G = \{j \in J: \ a_j > b\}$$

Proposition 1. The equation (1) is solvable if and only if $E \cup G \neq \emptyset$ and the interval solutions are the n-tuples $(X_1, \ldots, X_n)$ where for each $j \in J$ either

(i) there exists a $k \in G$ such that

$$X_j = \begin{cases} \{b\} & \text{if } j = k, \\ [0, b] & \text{if } j \in G \setminus \{k\}. \\ L & \text{if } j \in S \cup E, \end{cases}$$

or (ii) there exists a $k \in E$ such that

$$X_j = \begin{cases} [b, 1] & \text{if } j = k, \\ L & \text{if } j \in S \cup E \setminus \{k\}, \\ [0, b] & \text{if } j \in G. \end{cases}$$

The number of the interval solutions for (1) is equal to the cardinality of the set $E \cup G$ (which does not exceed n). We can compute the interval solutions of (1) in a polynomial time.

§ 3. Systems of linear equations

In this section systems of linear equations are solved by taking appropriate intersections of interval solutions of the single equations.

By $A \cdot X = B$ we denote the system of linear equations of the form

$$\begin{aligned} &(a_{11} \wedge x_1) \vee (a_{12} \wedge x_2) \vee \ldots \vee (a_{1n} \wedge x_n) = b_1 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &(a_{m1} \wedge x_1) \vee (a_{m2} \wedge x_2) \vee \ldots \vee (a_{mn} \wedge x_n) = b_m \end{aligned} \tag{2}$$

with coefficients $A = (a_{ij}) \in L^{I \times J}$, unknowns $X = (x_j) \in L^{J \times \{1\}}$ and constants $B = (b_i) \in L^{I \times \{1\}}$. We assume $|I| = m \in \mathbf{N}$, $|J| = n \in \mathbf{N}$.

The matrix $X^0 = (x_j^0) \in L^{J \times \{1\}}$ is a *point solution* of (2) if and only if $A \cdot X^0 = B$ holds. An n-tuple $(X_1, \ldots, X_n)$ with $X_i \subseteq L$ is an *interval solution* of (2) if each n-tuple $(x_1, \ldots, x_n)$ with $x_i \in X_i$ is a point solution of (2) and the n-tuple $(X_1, \ldots, X_n)$ is maximal with respect to this property.

The system (2) is *solvable* if it possesses at least one solution, otherwise it is not solvable or *unsolvable*.

The computing method for obtaining the interval solutions of (2) consists of the following.

For each $i \in I$ we denote by S_i, E_i, G_i the sets

$$S_i = \{j \in J\colon a_{ij} < b_i\}, \quad E_i = \{j \in J\colon a_{ij} = b_i\}, \quad G_i = \{j \in J\colon a_{ij} > b_i\}$$

According to Proposition 1 we can form the set V_i of the interval solutions of the i^{th} equality in (2):

$$V_i = \{X^{i,r_i}\colon X^{i,r_i} = (X_1^{i,r_i}, \ldots, X_n^{i,r_i}), \quad r_i \leqq |E_i \cup G_i|\}.$$

Let $r_i \leqq |E_i \cup G_i|$ be fixed for each $i \in I$. We denote by $X = (X_j)$, $j \in J$, an interval solution of (2) in which each X_j is the following nonempty intersection:

$$X_j = \bigcap_{i \in I} X_j^{i,r_i} \neq \emptyset \tag{3}$$

Let $V=\{X\colon X=(X_j),\ j\in J\}$ be the set of all interval solutions of (2). The elements of V are determined by (3). The cardinality of the set V is finite.

Using the above symbols we propose the following algorithm for computing the interval solutions of the system (2):

Algorithm 1

Step 1. For each $i\in I$ obtain the sets S_i, E_i, G_i, V_i.

Step 2. According to the expression (3) obtain the set V.

Proposition 2. The time complexity function of Algorithm 1 is exponential in the number of m equations.

Proof. It follows from the fact that the cardinality of the set V does not exceed the bound $\prod_{i=1}^{m} |E_i\cup G_i|\leqq n^m$.

In many cases we do not need all of the interval solutions of (2). For example, in fuzzy automata theory it is interesting whether the system is solvable or not and if the system is solvable — to compute one of its solutions. For this purpose we shall consider some of the properties of the system (2).

Two systems over L are called *equivalent* if each solution of the first one is a solution of the second and vice versa.

Proposition 3. If the system (2′) is obtained from the system (2) after a permutation of the equations then the systems (2) and (2′) are equivalent.

Let the system $A\cdot X=B$ be given. We denote by $(A\colon B)$ the matrix

$$(A\colon B)=\begin{pmatrix} a_{11}\ldots a_{1n}\colon\ b_1 \\ \ldots\ldots\ldots\ldots \\ a_{m1}\ldots a_{mn}\colon\ b_m \end{pmatrix}$$

We shall denote by (2′) the system $A'\cdot X=B'$:

$$\begin{aligned} &(a'_{11}\wedge x_1)\vee\ldots\vee(a'_{1n}\wedge x_n)=b'_1 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &(a'_{m1}\wedge x_1)\vee\ldots\vee(a'_{mn}\wedge x_n)=b'_m \end{aligned} \tag{2'}$$

obtained from (2) after a permutation of the equations in such a way that $b'_1\geqq \geqq b'_2\geqq\ldots\geqq b'_m$.

The systems (2) and (2′) are equivalent according to Proposition 3.

Let A^* be the matrix, obtained from the matrix A' by the rule

$$a^*_{ij}=\begin{cases} 0 & \text{if } a'_{ij}<b'_i, \\ b'_i & \text{if } a'_{ij}=b'_i, \\ 1 & \text{if } a'_{ij}>b'_i, \end{cases} \tag{4}$$

Proposition 4. The systems $A\cdot X=B$ and $A^*\cdot X=B'$ are equivalent.

Proof. According to Proposition 3 the systems $A\cdot X=B$ and $A'\cdot X=B'$ are equivalent. We shall prove that $A'\cdot X=B'\Leftrightarrow A^*\cdot X=B'$. If $A'\cdot X=B'$ then for

each $i \in I$

$$\exists j \in J\colon (a'_{ij} \geqq x_j = b'_i) \Leftrightarrow$$

$$\exists j \in J\colon (a'_{ij} \geqq b'_i \wedge x_j = b'_i) \vee (a'_{ij} = b'_i \wedge x_j \geqq b'_i) \Leftrightarrow$$

$$\exists j \in J\colon (a^*_{ij} = 1 \wedge x_j = b'_i) \vee (a^*_{ij} = b'_i \wedge x_j \geqq b'_i),$$

i.e. for each $i \in I$ there exists a $j \in J$ such that $a^*_{ij} \wedge x_j = b'_i$ and hence $A^* \cdot X = B'$.

Let $\underline{B} = \{b_1, \ldots, b_m\}$ be the set of the distinct elements in the matrix B of the system (2), resp. (2'). Having in mind the expression (4), it is clear that the elements a^*_{ij} of A^* and b'_i of B' belong to the set $\underline{B} \cup \{0, 1\}$.

Proposition 5. Let $X = (X_j)$ be an interval solution of the system $A^* \cdot X = B'$, where the components X_j, $j \in J$, are determined by (3). Each component X_j is among the following intervals: L, $[0, b_{p1}]$, $[b_{p2}, b_{p3}]$, $[b_{p4}, 1]$, where $b_{p1}, b_{p2}, b_{p3}, b_{p4} \in \underline{B}$.

The proof follows from Proposition 1 and the expression (3).

Corollary 1. Each interval solution of (2) has all its components among the following intervals: L, $[0, b_{p1}]$, $[b_{p2}, b_{p3}]$, $[b_{p4}, 1]$, where $b_{p1}, b_{p2}, b_{p3}, b_{p4} \in \underline{B}$.

Let B^n_m be the set of all n-fold variations with repetitions on the elements of the set $\underline{B}$.

Corollary 2. The system (2) is solvable if and only if there exists an $X^0 \in B^n_m$ such that $A \cdot X^0 = B$ holds.

Proof. If there exists an $X^0 \in B^n_m$ with $A \cdot X^0 = B$ then the system (2) is solvable. Conversely, if the system (2) is solvable, then each component X_j of an interval solution has the interval form determined by Corollary 1. Hence we can choose each component x^0_j of a point solution of (2) to be equal to an element of $\underline{B}$, i.e. $X^0 \in B^n_m$.

Having in mind Corollary 2 we propose the following algorithm for computing a point solution of the system (2), or for establishing its solvability.

Algorithm 2

Step 1. Find the set $\underline{B}$.
Step 2. Compute the set B^n_m.
Step 3. For each $X^0 \in B^n_m$ check whether it is a point solution of the system (2).
Step 4. List all point solutions determined in Step 3.
Step 5. If there exists no $X^0 \in B^n_m$ with $A \cdot X^0 = B$ then the system is unsolvable. Otherwise it is solvable and a set of point solutions is given in step 4.

Proposition 6. The time complexity function of Algorithm 2 is exponential in the number of n variables.

Proof. We can check whether $X^0 \in B^n_m$ is a solution of (2) in a polynomial time, but in a search problem manner. The set B^n_m is finite and $|B^n_m| = |\underline{B}|^n \leqq m^n$. Hence the Algorithm 2 is finite with exponential in the number of n variables time complexity function.

§ 4. A polynomial time algorithm

We propose a polynomial time algorithm for computing a point solution of the system (2) if it is solvable or for listing the numbers of the contradictory equations if the system is unsolvable.

In order to simplify the problem we introduce a symbol-matrix $\underline{A}$ with symbol-coefficients obtained from those of A^* if for each a_{ij}^* we put the corresponding type letter S, E or G (without index):

$$a_{ij} = \begin{cases} S & \text{if } a_{ij}^* = 0, \\ E & \text{if } a_{ij}^* = b_i', \\ G & \text{if } a_{ij}^* = 1. \end{cases} \tag{5}$$

The set of the solutions of the system (2) remains unchanged after this reduction step (5).

Let the system (2) be given and $X=(X_j)$ denote an interval solution of (2). Let the system (2′) and the matrix $\underline{A}$ be obtained. We assume $j \in J$ to be fixed in $\underline{A}$. In the following we denote by r the smallest number of the row with E-type coefficient in its j^{th} column and by k the greatest number of the row with G-type coefficient in its j^{th} column in $\underline{A}$.

In order to find a point solution of (2) we are interested in finding a point $x_j \in X_j$ with $a_{ij} \wedge x_j \leqq b_i$ for each $i \in I$. Especially we mark the i^{th} equation in a marker vector IND if $a_{ij} \wedge x_j = b_i$ holds.

Having in mind the above notions we obtain the following

Proposition 7. Let the system $A \cdot X = B$ be given.

i) if the j^{th} column in $\underline{A}$ contains a G-type coefficient then $x_j = b_k'$ implies $a_{ij}' \wedge x_j = b_i'$ for $i=k$ and for each $i>k$ with $a_{ij}' = b_i'$;

ii) if the j^{th} column in $\underline{A}$ does not contain any G-type coefficient but it contains an E-type coefficient then $x_j = b_r'$ implies $a_{ij}' \wedge x_j = b_i'$ for $i=r$ and for each $i>r$ with $a_{ij}' = b_i'$;

iii) if the j^{th} column in $\underline{A}$ does not contain neither G-type nor E-type coefficients then $a_{ij}' \wedge x_j < b_i'$ for each $x_j \in L$.

Proof. i) if $x_j = b_k'$ and $i=k$ then $a_{ij}' \wedge x_j = a_{kj}' \wedge b_k' = b_k'$ since $a_{kj}' > b_k'$; if $x_j = b_k'$, $i>k$ and $a_{ij}' = b_i'$ then $a_{ij}' = b_i' \leqq b_k'$ according to the order in (2′) implies $a_{ij}' \wedge x_j = b_i' \wedge b_k' = b_i'$;

ii) if $x_j = b_r'$ and $i=r$ then $a_{ij}' \wedge x_j = a_{rj}' \wedge b_r' = b_r' \wedge b_r' = b_r'$; if $x_j = b_r'$, $i>r$ and $a_{ij}' = b_i'$ then $a_{ij}' = b_i' \leqq b_r'$ according to the order in (2′) implies $a_{ij}' \wedge x_j = b_i' \wedge b_r' = b_i'$;

iii) if the j^{th} column in $\underline{A}$ contains only S-type coefficients then $a_{ij}' \wedge x_j \leqq a_{ij}' < b_i'$ for each $i \in I$ and arbitrary $x_j \in L$.

On this base we propose the following algorithm:

Algorithm 3

Step 1. Enter the matrix $(A:B)$.
Step 2. Form the matrix $\underline{A}$.
Step 3. Erase the marker vector IND.
Step 4. $j=0$.
Step 5. $j=j+1$.

Step 6. If $j>n$ go to 10.

Step 7. If the j^{th} column in $\underline{A}$ does not contain any G-type coefficient then go to 8. Otherwise $x_j=b'_k$. Put a mark in *IND* for $i=k$ and for each $i>k$ with $a'_{ij}=b'_i$. Put a mark in *IND* for each $i<k$ if $a'_{ij}\geqq b'_i=b'_k$. Go to Step 5.

Step 8. If the j^{th} column does not contain any E-type coefficient then go to step 9. Otherwise $x_j=b'_r$, put marks in *IND* for $i=r$ and for each $i>r$ with $a'_{ij}=b'_i$. Go to Step 5.

Step 9. $x_j=1$. Go to Step 5.

Step 10. If there exists at least one unmarked row in *IND* then the system is unsolvable and the unmarked equations are in contradiction with the marked ones. The marked equations form a compatible system. If all rows in *IND* are marked then the system is compatible and the components of the point solution $X=(x_j)$ are determined in Steps 7, 8, 9.

Theorem. The following problems are algorithmically decidable in polynomial time for the system (2):

i) whether the system is solvable or not;

ii) computing a point solution if the system is solvable;

iii) obtaining the numbers of the contradictory equations if the system is unsolvable.

The proof follows from Algorithm 3.

The program realisation of Algorithm 3 is available at the Center of Applied Mathematics in the Higher Institute for Mechanical and Electrical Engineering.

We shall consider two examples as a simple illustration of Algorithm 3.

Example 1. Solve the system

$$(0{,}3\wedge x_1)\vee(0{,}5\wedge x_2)\vee(0{,}4\wedge x_3)\vee(0{,}7\wedge x_4)=0{,}2$$

$$(0{,}8\wedge x_1)\vee(0{,}2\wedge x_2)\vee(0{,}7\wedge x_3)\vee(0{,}5\wedge x_4)=0{,}5$$

$$(0{,}2\wedge x_1)\vee(0{,}7\wedge x_2)\vee(0{,}5\wedge x_3)\vee(0{,}3\wedge x_4)=0{,}3$$

The (′)-system is

$$(0{,}8\wedge x_1)\vee(0{,}2\wedge x_2)\vee(0{,}7\wedge x_3)\vee(0{,}5\wedge x_4)=0{,}5$$

$$(0{,}2\wedge x_1)\vee(0{,}7\wedge x_2)\vee(0{,}5\wedge x_3)\vee(0{,}3\wedge x_4)=0{,}3$$

$$(0{,}3\wedge x_1)\vee(0{,}5\wedge x_2)\vee(0{,}4\wedge x_3)\vee(0{,}7\wedge x_4)=0{,}2$$

The matrix $(\underline{A}:B')$ and the marker vector *IND* are

$$(A:\ B')=\begin{pmatrix} G & S & G & E: & 0{,}5\\ S & G & G & E: & 0{,}3\\ G & G & G & G: & 0{,}2\end{pmatrix}\qquad IND=\begin{pmatrix}0\\0\\*\end{pmatrix}$$

The system is unsolvable. The contradictory equations have 0 in *IND*.

Example 2. Compute a point solution of the system

$$(0{,}2\wedge x_1)\vee\ 0{,}5\wedge x_2)\vee(0{,}7\wedge x_3)=0{,}4$$

$$(0{,}8\wedge x_1)\vee(0{,}2\wedge x_2)\vee(0{,}1\wedge x_3)=0{,}2$$

The matrix $(\underline{A}:B')$ and the marker vector IND are

$$(\underline{A}:\ B') = \begin{pmatrix} S & G & G:\ 0{,}4 \\ G & E & S:\ 0{,}2 \end{pmatrix} \quad IND = \begin{pmatrix} * \\ * \end{pmatrix}$$

The column vector $X=(0{,}2\ 0{,}4\ 0{,}4)^t$ is a point solution of this system.

I would like to express gratitude to prof. V. Trnkova and Dr. S. Ivanov for the valuable discussions and the interest in my work.

Abstract. A polynomial time algorithm for computing a point solution of a system of linear equations over a bounded chain is given.

References

[1] CUNINGHAME-GREEN, R. A., Minimax Algebra, Lecture Notes in Economics and Mathematical Systems, Vol. 166, Springer-Verlag, Berlin, 1979.
[2] EILENBERG, S., Automata, Languages and Machines, Vol. A, Acad. Press, New York, 1974.
[3] GAREY, M. R., JOHNSON, D. S., Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
[4] KUICH, W., Formal Power Series, Cycle-Free Automata and Algebraic Systems, Techn. Univ. Wien, Wien, 1982.
[5] MACLANE, S., BIRKHOFF, G., Algebra, Macmillan Publ. Co., New York, 1979.
[6] PEEVA, K. G., Linear Systems and Ordered Semirings, Proc. of the IX Summer School, Varna, 1983.
[7] SANTOS, E., On Reduction of Maxi-min Machines, J. Math. Anal. Appl., 40, 60—78 (1972).
[8] STARKE, P., Abstracte Automaten, Berlin, 1969.
[9] TOPENCHAROV, V. V., PEEVA, K. G., Equivalence, Reduction and Minimization of Finite Fuzzy Automata, J. Math. Anal. Appl., 84, 270—281 (1981).
[10] ZIMMERMANN, U., Linear and Combinatorial Optimization in Ordered Algebraic Structures, Amsterdam etc., North-Holl. 1981.

Received Oct. 11, 1984

SOFIA 1000
P.O. BOX 384
CENTER OF APPLIED MATHEMATICS

Metric representations by v_i-products

By F. GÉCSEG

The purpose of this paper is to compare the metric representation powers of the product and v_i-products introduced in [1]. It is shown that a class of automata is metrically complete with respect to the product if and only if it is metrically complete regarding the v_1-product. It is also proved that the v_3-product is metrically equivalent to the product.

We start with some basic notions and notations.

An *alphabet* is a nonvoid finite set. The free monoid generated by an alphabet X will be denoted by X^*. An element $p=x_1 \dots x_n \in X^*$ $(x_i \in X,\ i=1, \dots, n)$ is a *word* over X, and n is the *length* of p, in notation, $|p|=n$. If $n=0$ then p is the *empty word*, which will be denoted by e. For arbitrary integer $n(\geqq 0)$, $X^{(n)}$ will stand for the subset of X^* consisting of all words with length less than or equal to n.

An *automaton* is a system $\mathfrak{A}=(X, A, \delta)$, where X is the *input alphabet*, A is a nonvoid finite set of *states* and the mapping $\delta\colon A\times X \to A$ is the *transition function* of $\mathfrak{A}$. We extend δ to a mapping $\delta\colon A\times X^* \to A$ in the following way: for arbitrary $a\in A$, $\delta(a, e)=a$ and $\delta(a, px)=\delta(\delta(a, p), x)$ $(p\in X^*, x\in X)$.

Take an automaton $\mathfrak{A}=(X, A, \delta)$, a state $a\in A$ and an integer $n(\geqq 0)$. We say that the system $(\mathfrak{A}, a)$ is *n-free* if $\delta(a, p)\neq\delta(a, q)$ for arbitrary $p, q\in X^{(n)}$ with $p\neq q$.

If we add an output to an automaton then we get the concept of a sequential machine. More precisely, a system $\mathfrak{A}=(X, A, Y, \delta, \lambda)$ is a *Mealy machine*, where (X, A, δ) is an automaton, Y is the *output alphabet* and the mapping $\lambda\colon A\times X \to Y$ is the *output function* of $\mathfrak{A}$. We can extend λ to a mapping $\lambda\colon A\times X^* \to Y^*$ in the following way: for every $a\in A$, $\lambda(a, e)=e$ and $\lambda(a, px)=\lambda(a, p)\lambda(\delta(a, p), x)$. A mapping $\mu\colon X^* \to Y^*$ is called an *automaton mapping* if there exist a Mealy machine $\mathfrak{A}=(X, A, Y, \delta, \lambda)$ and an $a\in A$ such that $\mu(p)=\lambda(a, p)$ $(p\in X^*)$. If this is the case then we say that μ can be *induced* by $\mathfrak{A}$ in the state a.

Take a Mealy machine $\mathfrak{A}=(X, A, Y, \delta, \lambda)$, an automaton mapping $\mu\colon X^* \to Y^*$ and an integer $n(\geqq 0)$. It is said that $\mathfrak{A}$ *induces* μ *in length* n if for some $a\in A$, $\mu(p)=\lambda(a, p)$ $(p\in X^{(n)})$.

Let $\mathfrak{A}_j=(X_j, A_j, \delta_j)$ $(j=1, \dots, t)$ be automata, X and Y alphabets, and

$$\varphi\colon A_1\times \dots \times A_t\times X \to X_1\times \dots \times X_t,$$

$$\psi\colon A_1\times \dots \times A_t\times X \to Y$$

mappings. Then the Mealy machine $\mathfrak{A}=(X, A, Y, \delta, \lambda)$ is the *product (α_i-product, v_i-product)* of $\mathfrak{A}_j$ $(j=1, \ldots, t)$ with respect to X, Y and φ, ψ if the automaton (X, A, δ) is the product (α_i-product, v_i-product) of $\mathfrak{A}_j$ $(j=1, \ldots, t)$ with respect to X and φ, and for arbitrary $\mathbf{a}=(a_1, \ldots, a_t)\in A$ and $x\in X$, $\lambda(\mathbf{a}, x)=\psi(a_1, \ldots, a_t, x)$.

A class K of automata is *metrically complete* with respect to the product (α_i-product, v_i-product) if for arbitrary automaton mapping $\mu: X^*\to Y^*$ and integer $n(\geqq 0)$ there exists a product (α_i-product, v_i-product) $\mathfrak{A}=(X, A, Y, \delta, \lambda)$ of automata from K inducing μ in length n. Moreover, the v_i-product is *metrically equivalent* to the product provided that for every class K of automata and non-negative integer n an automaton mapping $\mu: X^*\to Y^*$ can be induced in length n by a v_i-product $\mathfrak{A}=(X, A, Y, \delta, \lambda)$ of automata from K if and only if it can be induced in length n by a product $\mathfrak{B}=(X, B, Y, \delta', \lambda')$ of automata from K.

Let $\mathfrak{A}_i=(X_i, A_i, \delta_i)$ $(i=1, \ldots, t)$ be automata, and take a product

$$\mathfrak{A} = (X, A, \delta) = \prod_{i=1}^{t} A_i[X, \varphi].$$

Then for arbitrary $\mathbf{a}=(a_1, \ldots, a_t)\in A$, $p\in X^*$ and i $(1\leqq i\leqq t)$ define $\varphi_i(\mathbf{a}, p)$ in the following way: $\varphi_i(\mathbf{a}, e)=e$ and $\varphi_i(\mathbf{a}, qx)=\varphi_i(\mathbf{a}, q)\varphi_i(\delta(\mathbf{a}, q), x)$ $(q\in X^*, x\in X)$.

For notions and notations not defined here, see [3] and [4].

Now we are ready to state and prove

Theorem 1. A class K of automata is metrically complete with respect to the product if and only if K is metrically complete with respect to the v_1-product.

Proof. The condition is obviously sufficient.

To show the necessity assume that K is metrically complete with respect to the product. We prove that for every alphabet Y and integer $k(\geqq 0)$ there exist a v_1-product $\mathfrak{D}=(Y, D, \delta'')$ of automata from K and a state $d\in D$ such that the system $(\mathfrak{D}, d)$ is k-free. This obviously implies that K is metrically complete with respect to the v_1-product.

It is shown in [2] that K is metrically complete with respect to the product if and only if for arbitrary integer $k(\geqq 0)$ there exist an $\mathfrak{A}=(X, A, \delta)$ in K a state $a_0\in A$ and a word $p\in X^*$ with $|p|=k$ such that $\delta(a_0, p)$ is ambiguous, that is $\delta(a_0, px)\neq\delta(a_0, px')$ for some $x, x'\in X$. Let us distinguish the following two cases.

Case 1. K contains an $\mathfrak{A}=(X, A, \delta)$ such that for certain pairwise distinct states $a_0, a_1, \ldots, a_{n-1}, a_1'$ and inputs $x_0, x_1, \ldots, x_{n-1}, x_1'$ we have

$$\delta(a_0, x_1) = a_1,\ \delta(a_1, x_2) = a_2, \ldots, \delta(a_{n-2}, x_{n-1}) = a_{n-1},\ \delta(a_{n-1}, x_0) = a_0$$

$$\text{and}\quad \delta(a_0, x_1') = a_1'.$$

Let $k(>0)$ be an integer, and take two words $p=y_1\ldots y_r y_{r+1}\ldots y_s$, $q=y_1\ldots y_r z_{r+1}\ldots$ $\ldots z_t\in Y^{(k)}$ $(y_1, \ldots, y_s, z_{r+1}, \ldots, z_t\in Y)$ with $t\leqq s$, and $y_{r+1}\neq z_{r+1}$ if $t\neq r$, where Y is an arbitrarily fixed alphabet. Consider the v_1-product

$$\mathfrak{B} = (Y, B, \delta') = \prod_{i=1}^{s+1} \mathfrak{B}_i[Y, \varphi, v]$$

given as follows.

$$\mathfrak{B}_i = \mathfrak{A} \quad (i = 1, \ldots, s+1).$$

$$\nu(1) = \emptyset \quad \text{and} \quad \nu(i) = i-1 \quad (i = 2, \ldots, s+1).$$

$$\varphi_i(a_j, y) = x_j \quad (i=2, \ldots, s+1;\ j=0, \ldots, n-1;\ y \in Y).$$

$$\varphi_i(a'_1, y) = \begin{cases} x_1 \text{ if } i = r+2 \text{ and } y = z_{r+1}, \\ x'_1 \text{ otherwise} \end{cases} \quad (i = 2, \ldots, s+1;\ y \in Y).$$

In all other cases φ is given arbitrarily such that the resulting product is a ν_1-product.

Take the state $\mathbf{b}=(b_1, b_2, \ldots, b_{s+1}) \in B$ with $b_1=a'_1$, $b_i=a_{n-(i-2)}$ $(i=2, \ldots, s+1)$, where the indices of a's are taken modulo n. One can easily show by induction on j that for every $j(=1, \ldots, s)$

$$\delta'(\mathbf{b}, y_1 \ldots y_j) = (c_1, \ldots, c_{j+1}, c_{j+2}, \ldots, c_{s+1})$$

where $c_{j+1}=a'_1$ and $c_i=a_{n-(i-2)+j}$ $(i=j+2, \ldots, s+1)$. Moreover, for every $j(=r+1, \ldots, t)$

$$\delta'(\mathbf{b}, y_1 \ldots y_r z_{r+1} \ldots z_j) = (c_1, \ldots, c_{j+1}, \ldots, c_{s+1})$$

where $c_i=a_{n-(i-2)+j}$ $(i=j+1, \ldots, s+1)$. (The indices of a's are considered modulo n in the latter two cases, too.)

Therefore, the last component of $\delta'(\mathbf{b}, p)$ is a'_1, and the last component of $\delta'(\mathbf{b}, q)$ is in the set $\{a_0, a_1, \ldots, a_{n-1}\}$. Thus $\delta'(\mathbf{b}, p) \neq \delta'(\mathbf{b}, q)$.

Case 2. K does not satisfy the conditions of Case 1. Then for every integer $k(\geqq 0)$ there is an $\mathfrak{A}=(X, A, \delta)$ in K with pairwise distinct states $a_0, a_1, \ldots, a_k, a_{k+1}, a'_{k+1}$ and inputs $x_1, x_2, \ldots, x_k, x_{k+1}, x'_{k+1}$ such that $\delta(a_i, x_{i+1})=a_{i+1}$ $(i=0, \ldots, k)$ and $\delta(a_k, x'_{k+1})=a'_{k+1}$. Again take the alphabet Y and the words p, q of Case 1. Consider the ν_1-product

$$\mathfrak{B} = (Y, B, \delta') = \prod_{i=1}^{s} \mathfrak{B}_i[Y, \varphi, \nu]$$

given in the following way.

$$\mathfrak{B}_i = \mathfrak{A} \quad (i = 1, \ldots, s).$$

$$\nu(1) = \emptyset \quad \text{and} \quad \nu_i = i-1 \quad (i = 2, \ldots, s).$$

$$\varphi_1(y_1) = x_{k+1} \quad (\text{and} \quad \varphi_1(z_1) = x'_{k+1} \quad \text{if} \quad r = 0 \quad \text{and} \quad t \neq 0).$$

$$\varphi_i(a_j, y) = \begin{cases} x'_{k+1} \quad \text{if} \quad i = r+1, \quad j = k+1 \quad \text{and} \quad y = z_{r+1}, \\ x_j \quad \text{otherwise} \end{cases}$$

$$(i = 2, \ldots, s; \quad j = 1, \ldots, k+1).$$

$$\varphi_i(a'_{k+1}, y) = x'_{k+1} \quad (i=2, \ldots, s).$$

In all other cases φ is given arbitrarily in accordance with the definition of the ν_1-product.

Take the state $\mathbf{b}=(a_k, a_{k-1}, \ldots, a_{k-s+1}) \in B$. Again it is easy to show that for every $j(=1, \ldots, s)$

$$\delta'(\mathbf{b}, y_1 \ldots y_j) = (c_1, \ldots, c_j, \ldots, c_s),$$

where $c_i=a_{k-(i-1)+j}$ $(i=j, \ldots, s)$. Moreover, for every $j(=r+1, \ldots, s)$

$$\delta'(\mathbf{b}, y_1 \ldots y_r z_{r+1} \ldots z_j) = (c_1, \ldots, c_j, \ldots, c_s)$$

with $c_j=a'_{k+1}$ and $c_i=a_{k-(i-1)+j}$ $(i=j+1, \ldots, s)$.

Therefore, the last component of $\delta'(\mathbf{b}, p)$ is a_{k+1}. If $s=t$ then the last component of $\delta'(\mathbf{b}, q)$ is a'_{k+1}. Moreover, if $t<s$ then the last component of $\delta'(\mathbf{b}, q)$ is $a_{k-(s-1)+t}$. In both cases we have $\delta'(\mathbf{b}, p) \neq \delta'(\mathbf{b}, q)$.

To end the proof of Theorem 1 take an integer $k(\geqq 1)$ and an alphabet Y. Moreover, set $I=\{(p, q) | p, q \in Y^{(k)}, p \neq q\}$. As it has been shown for every pair $(p, q) \in I$ there exist a v_1-product $\mathfrak{D}_{(p,q)}=(Y, D_{(p,q)}, \delta_{(p,q)})$ of automata from K and a state $d_{(p,q)} \in D_{(p,q)}$ such that $\delta_{(p,q)}(d_{(p,q)}, p) \neq \delta_{(p,q)}(d_{(p,q)}, q)$. Form the direct product $\mathfrak{D}=\Pi(\mathfrak{D}_{(p,q)} | (p, q) \in I)$, and take the state $\mathbf{d} \in D$ with $pr_{(p,q)}(\mathbf{d})=d_{(p,q)}$, where $pr_{(p,q)}$ denotes the $(p, q)^{\text{th}}$ projection. Obviously, $(\mathfrak{D}, \mathbf{d})$ is a k-free system. Since the direct product of v_1-products of automata is isomorphic to a v_1-product of the same automata this completes the proof of Theorem 1.

Let us note that the v_1-product used in the proof of Theorem 1 is also an α_0-product.

Next we prove

Theorem 2. The product is metrically equivalent to the v_3-product.

Proof. Let K be a class of automata. If K is metrically complete with respect to the product then, by Theorem 1, for arbitrary integer $k(\geqq 0)$ every automaton mapping $\mu: X^* \to Y^*$ can be induced in length $k+1$ by a v_1-product $\mathfrak{A}=(X, A, Y, \delta, \lambda)$ of automata from K. Thus we assume that K is not metrically complete with respect to the product. Therefore, none of Case 1 and Case 2 holds for K. This implies that either there is no ambiguous state in any of the automata from K or there is a maximal positive integer k such that for some $\mathfrak{A}=(X, A, \delta) \in K$, $a \in A$ and $p \in X^*$ with $|p|=k-1$, $\delta(a, p)$ is ambiguous. In the first case every product of automata from K can be given as a quasi-direct product of the same automata. Thus we suppose the existence of the above k.

Let

$$\mathfrak{A} = (X, A, \delta) = \prod_{i=1}^{s} \mathfrak{A}_i[X, \varphi] \quad (\mathfrak{A}_i = (X_i, A_i, \delta_i) \in K, \ i = 1, \ldots, s)$$

be a product and $\mathbf{a}=(a_1, \ldots, a_s) \in A$ a state. We shall prove the existence of a v_3-product

$$\mathfrak{B} = (X, B, \delta') = \prod_{i=1}^{t} \mathfrak{B}_i[X, \varphi', \nu] \quad (\mathfrak{B}_i = (X'_i, B_i, \delta'_i), \ i = 1, \ldots, t)$$

with a state $\mathbf{b}=(b_1, \ldots, b_t) \in B$ such that the following conditions are satisfied.

(i) $(\mathfrak{B}_1, b_1)$ is k-free, $X'_1=X$, φ'_1 is the identity mapping on X and $\mathfrak{B}_1$ is a v_1-product of automata from K.

(ii) $\mathfrak{B}_2$ is a v_1-product of automata from K, $X'_2=X$ and for any two words $p, q \in X^*$ with $|p|<k$ and $|q| \geqq k$, $\delta'_2(b_2, \varphi'_2(\mathbf{b}, p)) \neq \delta'_2(b_2, \varphi'_2(\mathbf{b}, q))$.

(iii) $\mathfrak{B}_i \in K$ $(i=3, \ldots, t)$.

(iv) For arbitrary two words $p, q \in X^*$ with $|p|=|q|=k$ and integer i $(1 \leqq i \leqq s)$ there is a j $(1 \leqq j \leqq t)$ with $\mathfrak{B}_j=\mathfrak{A}_i$, $b_j=a_i$, $\delta'_j(b_j, \varphi'_j(\mathbf{b}, p))=\delta_i(a_i, \varphi_i(\mathbf{a}, p))$ and $\delta'_j(b_j, \varphi'_j(\mathbf{b}, q))=\delta_i(a_i, \varphi_i(\mathbf{a}, q))$.

This will imply that the subautomaton of $\mathfrak{A}$ generated by $\mathbf{a}$ is a homomorphic image of the subautomaton of $\mathfrak{B}$ generated by $\mathbf{b}$. Indeed, take two words $p, q \in X^*$ with $\delta(\mathbf{a}, p) \neq \delta(\mathbf{a}, q)$. It is enough to show that $\delta'(\mathbf{b}, p) \neq \delta'(\mathbf{b}, q)$. Let us distinguish the following cases.

(I) $|p|, |q| \leqq k$. Then $\delta'(\mathbf{b}, p) \neq \delta'(\mathbf{b}, q)$ since they differ at least in their first components.

(II) $|p| < k$ and $|q| > k$. Then $\delta'(\mathbf{b}, p)$ and $\delta'(\mathbf{b}, q)$ are different at least in their 2nd components.

(III) $|p|, |q| \geqq k$. First of all observe that, by the maximality of k, for arbitrary automaton $\mathfrak{C} = (Y, C, \delta'') \in K$, state $c \in C$ and words $r, r_1, r_2 \in Y^*$ with $|r| = k$ and $|r_1| = |r_2|$, $\delta''(c, rr_1) = \delta''(c, rr_2)$. Let $p = p_1 p_2$ and $q = q_1 q_2$ ($|p_1| = |q_1| = k$). Moreover, let i ($1 \leqq i \leqq s$) be an index for which $\delta_i(a_i, \varphi_i(\mathbf{a}, p)) \neq \delta_i(a_i, \varphi_i(\mathbf{a}, q))$. Take the index j given by (iv) to this i and p_1, q_1. Then by our remark above $\delta'_j(b_j, \varphi'_j(\mathbf{b}, p_1 p_2)) = \delta'_j(b_j, \varphi'_j(\mathbf{b}, p_1) p'_2) = \delta_i(a_i, \varphi_i(\mathbf{a}, p_1) p'_2) = \delta_i(a_i, \varphi_i(\mathbf{a}, p_1 p_2))$ where $p'_2 \in X_i^*$ is a word with $|p'_2| = |p_2|$. Similarly, $\delta'_j(b_j, \varphi'_j(\mathbf{b}, q_1 q_2)) = \delta_i(a_i, \varphi_i(\mathbf{a}, q_1 q_2))$. Therefore, $\delta'(\mathbf{b}, p) \neq \delta'(\mathbf{b}, q)$ since they differ at least in their j^{th} components.

The k-free automaton in (i) can be constructed by using the same method as in the proof of Theorem 1 (according to Case 2).

To give $\mathfrak{B}_2$ take an automaton $\mathfrak{C} = (Y, C, \delta'') \in K$ with pairwise distinct states $c_0, c_1, \ldots, c_{k-1}, c_k, c'_k$ and inputs $y_1, \ldots, y_{k-1}, y_k, y'_k$ such that $\delta''(c_0, y_1) = c_1, \ldots, \delta''(c_{k-2}, y_{k-1}) = c_{k-1}$, $\delta''(c_{k-1}, y_k) = c_k$ and $\delta''(c_{k-1}, y'_k) = c'_k$. Form the single factor v_1-product

$$\mathfrak{B}_2 = \mathfrak{C}[X, \varphi'', \nu']$$

where $\nu'(1) = 1$ and $\varphi''(c_i, x) = y_{i+1}$ ($i = 0, \ldots, k-1$; $x \in X$). Moreover, in all other cases φ'' is given arbitrarily. Since K is not metrically complete $\mathfrak{B}_2$ satisfies (ii).

Next we show that for arbitrary words $p, q \in X^*$ with $|p| = |q| = k$ and integer i ($1 \leqq i \leqq s$) there are a v_3-product

$$\mathfrak{D} = (X, D, \delta'') = \prod_{i=1}^{r} \mathfrak{C}_i[X, \varphi'', \nu']$$

($\mathfrak{C}_i = (Y_i, C_i, \delta''_i) \in K$, $i = 1, \ldots, r$) and a state $\mathbf{d} = (d_1, \ldots, d_r) \in D$ such that $\mathfrak{C}_r = \mathfrak{A}_i$, $d_r = a_i$, $\delta''_r(d_r, \varphi''_r(\mathbf{d}, p)) = \delta_i(a_i, \varphi_i(\mathbf{a}, p))$ and $\delta''_r(d_r, \varphi''_r(\mathbf{d}, q)) = \delta_i(a_i, \varphi_i(\mathbf{a}, q))$. Then taking the direct product of $\mathfrak{B}_1$, $\mathfrak{B}_2$ and these automata $\mathfrak{D}$ the resulting automaton $\mathfrak{B}$ with a suitable $\mathbf{b} \in B$ will obviously satisfy (i)—(iv).

Since the case

$$(*) \qquad \delta_i(a_i, \varphi_i(\mathbf{a}, p)) = \delta_i(a_i, \varphi_i(\mathbf{a}, q))$$

is trivial we may assume that $(*)$ does not hold. Then $p \neq q$. Let $p = x_1 \ldots x_m x_{m+1} \ldots x_k$, $q = x_1 \ldots x_m y_{m+1} \ldots y_k$, $x_{m+1} \neq y_{m+1}$, $\varphi_i(\mathbf{a}, p) = \bar{p} = u_1 \ldots u_m u_{m+1} \ldots u_k$ and $\varphi_i(\mathbf{a}, q) = \bar{q} = u_1 \ldots u_m v_{m+1} \ldots v_k$. Moreover, set $p_j = x_1 \ldots x_j$; $\bar{p}_j = u_1 \ldots u_j$ ($j = 0, 1, \ldots, k$) and

$$q_j = \begin{cases} x_1 \ldots x_j & \text{if } 0 \leqq j \leqq m, \\ x_1 \ldots x_m y_{m+1} \ldots y_j & \text{if } m < j \leqq k, \end{cases}$$

$$\bar{q}_j = \begin{cases} u_1 \ldots u_j & \text{if } 0 \leqq j \leqq m, \\ u_1 \ldots u_m v_{m+1} \ldots v_j & \text{if } m < j \leqq k. \end{cases}$$

Denote a_i by c_0. Let l_1 be the smallest integer u for which there is a v with $u < v \leqq k$ such that $\delta_i(c_0, \bar{p}_u) = \delta_i(c_0, \bar{p}_v)$. If there are no such u and v then let $l_1 = k$. Sim-

ilarly, let l_2 be the least integer u such that for some v $(u<v\leqq k)$, $\delta_i(c_0, \bar{q}_u)=\delta_i(c_0, \bar{q}_v)$. Again if there are no such u and v then let $l_2=k$. Assume that $l_1\geqq l_2$. Finally, denote by w the maximal number with $\delta_i(c_0, \bar{p}_w)=\delta_i(c_0, \bar{q}_w)$ $(0\leqq w\leqq k)$. Since $\delta_i(c_0, \bar{p})\neq\delta_i(c_0, \bar{q})$ the inequality $l_2>w$ holds. Moreover, $w\geqq m$. Let us introduce the notations $\delta_i(c_0, \bar{p}_j)=c_j$ $(j=0, \ldots, l_1)$ and $\delta_i(c_0, \bar{q}_j)=c'_j$ $(j=0, \ldots, l_2)$. Then the elements $c_0, \ldots, c_w, c_{w+1}, c'_{w+1}$ are pairwise distinct, and so are the elements of the sets $\{c_0, \ldots, c_{l_1}\}$ and $\{c'_0, \ldots, c'_{l_2}\}$. We continue the proof by distinguishing the following two cases.

Case 1. $w=m$. Then let $r=2$ and $\mathfrak{C}_1=\mathfrak{C}_2=\mathfrak{A}_i$. Moreover, $\nu'(1)=1$, $\nu'(2)=\{1, 2\}$ and

$$\varphi''_1(c_j, x) = u_{j+1} \quad (j = 0, \ldots, l_1-1;\ x\in X),$$
$$\varphi''_2(c_j, c_j, x_{j+1}) = u_{j+1} \quad (j = 0, \ldots, l_1-1),$$
$$\varphi''_2(c_j, c'_j, y_{j+1}) = v_{j+1} \quad (j = m, \ldots, l_2-1).$$

In all other cases φ'' is given arbitrarily. φ'' is well defined. It is obvious that φ''_1 is a function. Assume that $(c_j, c_j, x_{j+1})=(c_j, c'_j, y_{j+1})$ holds for some j $(m<j<l_2)$. But this would imply $w>m$.

It is seen immediately that by taking $\mathbf{d}=(c_0, c_0)$ the equalities

$$\delta''(\mathbf{d}, p_j) = (c_j, c_j) \quad (j = 0, \ldots, l_1)$$

and

$$\delta''(\mathbf{d}, q_j) = (c_j, c'_j) \quad (j = 0, \ldots, l_2)$$

hold. Since K is not metrically complete with respect to the product, by the choice of l_1 and l_2, this implies

$$\delta''(\mathbf{d}, p) = (c, \delta_i(c_0, \bar{p})) \quad (c\in A_i)$$

and

$$\delta''(\mathbf{d}, q) = (c', \delta_i(c_0, \bar{q})) \quad (c'\in A_i).$$

Case 2. $w>m$. Let $r=w-m+2$ and $\mathfrak{C}_1=\ldots=\mathfrak{C}_r=\mathfrak{A}_i$. Moreover, $\nu'(1)=1$, $\nu'(j)=j-1$ $(j=2, \ldots, r-2)$, $\nu'(r-1)=r-1$ and $\nu'(r)=\{r-2, r-1, r\}$. Furthermore,

$$\varphi''_1(c_{w-m+l}, x_{l+1}) = u_{w-m+l+1} \quad (l = 0, \ldots, m),$$
$$\varphi''_1(c_w, y_{m+1}) = v_{w+1},$$
$$\varphi''_j(c_{w-m-j+2+l}, x_{l+1}) = u_{w-m-j+2+l} \quad (j = 2, \ldots, r-2;\quad l = 0, \ldots, m+j-1),$$
$$\varphi''_j(c_{w-m-j+2+l}, y_{l+1}) = u_{w-m-j+2+l} \quad (j = 2, \ldots, r-2;\quad l = m, \ldots, m+j-2),$$
$$\varphi''_j(c'_{w+1}, y_{m+j}) = v_{w+1} \quad (j = 2, \ldots, r-2),$$
$$\varphi''_{r-1}(c_l, x_{l+1}) = u_{l+1} \quad (l = 0, \ldots, l_1-1),$$
$$\varphi''_{r-1}(c_l, y_{l+1}) = u_{l+1} \quad (l = m, \ldots, l_2-1),$$
$$\varphi''_r(c_{l+1}, c_l, c_l, x_{l+1}) = u_{l+1} \quad (l = 0, \ldots, w),$$
$$\varphi''_r(c_{l+1}, c_l, c_l, y_{l+1}) = u_{l+1} \quad (l = m, \ldots, w-1),$$
$$\varphi''_r(c'_{w+1}, c_w, c_w, y_{w+1}) = v_{w+1},$$
$$\varphi''_r(c, c_{w+l}, c_{w+l}, x_{w+l+1}) = u_{w+l+1} \quad (c\in A_i,\ l = 1, \ldots, l_1-(w+1)),$$
$$\varphi''_r(c, c_{w+l}, c'_{w+l}, y_{w+l+1}) = v_{w+l+1} \quad (c\in A_i,\ l = 1, \ldots, l_2-(w+1)).$$

In all other cases φ'' is given arbitrarily in accordance with the definition of the v_3-product. φ'' is well defined. This is clear in all cases except when

$$(c, c_{w+l}, c_{w+l}, x_{w+l+1}) = (c', c_{w+l}, c'_{w+l}, y_{w+l+1})$$

for an l $(1 \leqq l \leqq l_2-(w+1))$. But this would contradict the choice of w.

One can easily show by induction on l that for $\mathbf{d}=(c_{w-m}, c_{w-m-1}, \ldots, c_1, c_0, c_0)$ the following equalities hold.

$$\delta''(\mathbf{d}, p_l) = (c_{w-m+l}, c_{w-m-1+l}, \ldots, c_{1+l}, c_l, c_l) \quad (l = 0, \ldots, m),$$

$$\delta''(\mathbf{d}, p_{m+l}) = (c''_1, \ldots, c''_{l-1}, c_{w+1}, c_w, \ldots, c_{m+l+1}, c_{m+l}, c_{m+l})$$
$$(c''_1, \ldots, c''_{l-1} \in A_i;\ l = 1, \ldots, w-m),$$

$$\delta''(\mathbf{d}, q_{m+l}) = (c''_1, \ldots, c''_{l-1}, c'_{w+1}, c_w, \ldots, c_{m+l+1}, c_{m+l}, c_{m+l})$$
$$(c''_1, \ldots, c''_{l-1} \in A_i;\ l = 1, \ldots, w-m),$$

$$\delta''(\mathbf{d}, p_l) = (c''_1, \ldots, c''_{r-2}, c_l, c_l) \quad (c''_1, \ldots, c''_{r-2} \in A_i;\ l = w+1, \ldots, l_1),$$

$$\delta''(\mathbf{d}, q_l) = (c''_1, \ldots, c''_{r-2}, c_l, c'_l) \quad (c''_1, \ldots, c''_{r-2} \in A_i;\ l = w+1, \ldots, l_2).$$

Since K is not metrically complete with respect to the product, by the choice of l_1 and l_2, the last two equalities imply

$$\delta''(\mathbf{d}, p) = (c''_1, \ldots, c''_{r-1}, \delta_i(c_0, \bar{p})) \quad \text{and} \quad \delta''(\mathbf{d}, q) = (\bar{c}_1, \ldots, \bar{c}_{r-1}, \delta_i(c_0, \bar{q}))$$
$$(c''_1, \ldots, c''_{r-1}, \bar{c}_1, \ldots, \bar{c}_{r-1} \in A_i)$$

which ends the proof of Theorem 2.

Let us note that the v_3-product $\mathfrak{B}$ in the proof of Theorem 2 is also an α_1-product.

DEPT. OF COMPUTER SCIENCE
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H—6720

References

[1] DÖMÖSI, P., B. IMREH, On v_i-products of automata, Acta Cybernet., v. VI. 2, 1983, pp. 149—162.
[2] Гечег, Ф., Метрически полные системы автоматов, Кибернетика, №3, 1968, pp. 96—98.
[3] GÉCSEG, F., Representation of automaton mappings in finite length, Acta Cybernet., v. II. 4, 1975, pp. 285—289.
[4] GÉCSEG, F., On v_1-products of commutative automata, Acta Cybernet., v. VII. 1, 1985, pp. 55—59.

Received June 14, 1984

A partial solution of the finite spectrum problem

P. ECSEDI-TÓTH

1. Introduction

Let φ be an arbitrary first order sentence. By the finite spectrum of φ, Sp φ, the following set is meant:

$$\text{Sp } \varphi = \{n \in \omega | (\exists \underline{A})\ (\text{card } A = n \text{ and } \underline{A} \models \varphi)\}.$$

The finite spectrum problem, due to Scholz [9], can be paraphrased in some different ways. Here are four possibilities:

FSP1 ([3], p. 512). Given φ arbitrarily, is there a first order sentence ψ such that $(\forall n \in \omega)\ (n \in \text{Sp } \varphi \Leftrightarrow n \notin \text{Sp } \psi)$?

FSP2 ([7], p. 269). Given φ arbitrarily, characterize Sp φ as a set of positive integers.

FSP3. Let $N \subset \omega$ be arbitrary. Is there a first order sentence φ such that Sp $\varphi = N$?

FSP4. Characterize those subsets of ω which are the finite spectra of first order sentences.

The finite spectrum problem, beyond its historical interest, is closely related to some recent problems in theoretical computer science concerning *NP* completeness [5], [6]. Albeit a solution would be very useful, no general answer is known at the moment. For many interesting partial solutions, expecially for FSP3, see [2], [8]. As far as we know, however, no syntactically characterized non-trivial class of first order sentences has been given for which the finite spectrum problem is solvable. The purpose of the present paper is to provide one such class, the class of equality-free first order sentences, and to give answers to all of the four questions FSP 1—4 for this particular class.

We hope, that our considerations can help to attack the general problem by indicating where difficulties arise.

The class of equality-free first order sentences is by no means trivial. Indeed, it is known that computability can be formalized in a fragment of equality-free sentences (by equality-free universal Horn sentences, cf. [1], for a proof), which,

in turn, play an essential role in PROLOG programming and thus, in the fifth-generation computer projects. Hence, the results presented here can have some impacts on the problems of complexity theory connected to the finite spectrum problem too.

2. Notations

Structures will be denoted by underlined capitals $\underline{A}$, $\underline{B}$; the corresponding capitals without underlining A, B stand for the universes of $\underline{A}$, $\underline{B}$, respectively. Constant, relation and function symbols are written in the lower case letters c, r, f; while their realizations in a structure, say in $\underline{A}$, will be denoted by $C^{(A)}$, $R^{(A)}$ and $F^{(A)}$.

We may suppose that there are only finitely many symbols in the first order language since our particular topic concerns simultaneously only finitely many sentences. For the sake of convenience, we shall assume that there are constant symbols, relation symbols and function symbols in the language. Thus, the universe of any structure is nonvoid. Therefore, we shall be interested in the variants of FSP 1—4 where ω is replaced by $[\omega]=\{1, 2, \ldots\}$. In the sequel, we shall always mean these versions when we are speaking on the finite spectrum problem.

3. An upward Löwenheim—Skolem theorem

Theorem 1. Let $\underline{A}$ be any structure of cardinality $n\in[\omega]$. Then, there is a structure $\underline{B}$ such that the cardinality of $\underline{B}$ is $n+1$ and $\underline{B}$ is elementarily equivalent to $\underline{A}$ in the equality-free sense.

Proof. Let b be a new element and define $B=A\cup\{b\}$. Let $h\colon B\to A$ be any onto mapping such that the restriction of h to A is the identity. Define $\underline{B}$ by the following items.

(i) For every constant symbol c, let $C^{(B)}=C^{(A)}$.

(ii) For every function symbol f of arity m and for arbitrary elements $b_1, \ldots, b_m\in B$, put

$$F^{(B)}(b_1, \ldots, b_m) = F^{(A)}(h(b_1), \ldots, h(b_m)).$$

(iii) For every relation symbol r of arity m and elements $b_1, \ldots, b_m\in B$, let

$$\langle b_1, \ldots, b_m\rangle\in R^{(B)} \quad \text{iff} \quad \langle h(b_1), \ldots, h(b_m)\rangle\in R^{(A)}.$$

It is easily seen, that h is a homomorphism from $\underline{B}$ onto $\underline{A}$ in the algebraic sense and h preserves relations. It follows, that the kernel of h is a congruence on $\underline{B}$ which is invariant over relations; hence $\underline{B}$ is correctly defined.

We shall prove by a straight-forward induction that $\underline{B}$ is elementarily equivalent to $\underline{A}$ in the equality-free sense. More precisely, we prove:

For arbitrary equality-free formula φ and assignment $k\colon V\to B$ (where V is the set of variables)

$$B\models\varphi[k] \quad \text{iff} \quad A\models\varphi[k_h] \tag{1}$$

where $k_h\colon V\to A$ is defined by $k_h(v)=h(k(v))$.

First we notice, that for any term t,

$$h(t^{(B)}[k]) = t^{(A)}[k_h]. \tag{2}$$

(Here, e.g. $t^{(A)}[k_h]$ stands for the familiar notion "the value of t in $\underline{A}$ at k_h".) Indeed, if t is a variable or a constant symbol, then (2) trivially holds by definition. Let t be of the form $f(t_1, \ldots, t_m)$ and assume that (2) is true for t_i $(1 \leqq i \leqq m)$. Then,

$$\begin{aligned} h(t^{(B)}[k]) &= h\big(F^{(B)}(t_1^{(B)}[k], \ldots, t_m^{(B)}[k]\big) = \\ &\overset{\text{(ii)}}{=} h\big(F^{(A)}(h(t_1^{(B)}[k]), \ldots, h(t_m^{(B)}[k])\big) = \\ &\overset{(*)}{=} h\big(F^{(A)}(t_1^{(A)}[k_h], \ldots, t_m^{(A)}[k_h])\big) = \\ &\overset{(**)}{=} F^{(A)}(t_1^{(A)}[k_h], \ldots, t_m^{(A)}[k_h]) = t^{(A)}[k_h] \end{aligned}$$

(where the equalities denoted by (*) and (**) hold by the induction hypothesis and since h is the identity on A, respectively). Hence (2) is true.

Turning to the proof of (1), if φ is an equality-free prime formula of the form $r(t_1, \ldots, t_m)$, then

$$B \models r(t_1, \ldots, t_m)[k] \quad \text{iff} \quad A \models r(t_1, \ldots, t_m)[k_h]$$

is easily seen by using (iii) and (2).

The induction trivially passes over negation and conjunction.

Let φ be an equality-free formula of the form $\exists v \psi$, and suppose, that (1) holds for ψ. Then,

$\underline{B} \models \exists v \psi[k]$ iff

There is an assignment k': $V \to B$ such that (3)

$k(w) = k'(w)$ provided $w \neq v$ and $\underline{B} \models \psi[k']$.

Similarly,

$\underline{A} \models \exists v \psi[k_h]$ iff

There is an assignment k'_h: $V \to A$ such that (4)

$k_h(w) = k'_h(w)$ if $w \neq v$ and $\underline{A} \models \psi[k'_h]$.

By the induction hypothesis, (3) implies (4) for the assignment k'_h, defined by $k'_h(v) = h(k'(v))$, $k'_h(w) = h(k(w))$ if $w \neq v$. Similarly, if (4) holds, then, since h is onto, there is an assignment k' such that $k'(w) = k(w)$ if $v \neq w$ and $h(k'(v)) = k'_h(v)$ and $\underline{B} \models \psi[k']$. Hence (3) and (4) are equivalent and thus,

$$B \models \exists v \psi[k] \quad \text{iff} \quad A \models \exists v \psi[k_h].$$

This completes the induction. It follows, that $\underline{B}$ is elementarily equivalent to $\underline{A}$ in the equality-free sense.

Q.E.D.

Corollary 2. Let $\underline{A}$ be any structure of cardinality n $(n \in [\omega])$. Then for every $m \in [\omega]$, $m \geqq n$, there exists a structure $\underline{B}$ such that B has cardinality m and A is elementarily equivalent to B in the equality-free sense.

Proof. Iterate Theorem 1 $m-n$ times.

Q.E.D.

Remark. Corollary 2 can be considered as a sharpened version of the (finitary) upward Löwenheim—Skolem theorem for equality-free languages. For some generalization see [4].

The following assertion indicates, that the downward Löwenheim—Skolem theorem has no similar sharpening. Besides, it has an application in the next section.

Theorem 3. (i) For every $n\in[\omega]$, there exists a set T_n of equality-free sentences such that T_n has a model of cardinality m iff $m\geqq n$.

(ii) There exists a set T_ω of equality-free sentences such that T_ω has only infinite models.

Proof. Without loss of generality, we may suppose that there is a unary function symbol f and there is a unary relation symbol r in the language. Let c be a constant symbol.

(i) We shall use the following notation: for $k\in\omega, f^{(0)}(c)=c, f^{(k+1)}(c)=f(f^{(k)}(c))$. Let $n\in[\omega]$. We set

$$T_n=\{r(c)\}\cup\{\neg r(f^{(k)}(c))\,|\,1\leqq k<n-1\}\cup\{r(f^{(n-1)}(c))\}.$$

A trivial induction shows that if $1\leqq m<n$ then no model of T_n exists with cardinality m. On the other hand, it is easy to construct a structure $\underline{A}$ such that A has cardinality n and $\underline{A}\models T_n$. It follows by Corollary 2 that for each $m\geqq n$, T_n has a model with cardinality m.

(ii) The following additional notations will be used: for $k\in\omega$, $\neg_0$ is the empty sequence, $\neg_{k+1}=\langle\neg,\langle\neg_k\rangle\rangle$; and $|\log_2 k|=\max\{m|m\in\omega \text{ and } 2^m\leqq k\}$. Let

$$T_\omega=\{\neg_{|\log_2 k|}r(f^{(k-1)}(c))\,|\,k\in[\omega]\}.$$

Again, it is easily seen by induction that T_ω has no finite models, but a model of T_ω with cardinality ω is easy to obtain.

Q.E.D.

4. The finite spectrum of equality-free sentences

Corollary 4. Let φ be an arbitrary equality-free first order sentence. Then,

(i) For every $n\in[\omega]$, if $n\in\mathrm{Sp}\,\varphi$, then for all $m\in\omega$, $m\geqq n$ implies that $m\in\mathrm{Sp}\,\varphi$.

(ii) For any equality-free first order sentence ψ, if $\mathrm{Sp}\,\varphi\neq\emptyset$ and $\mathrm{Sp}\,\psi\neq\emptyset$ then $\mathrm{Sp}\,\varphi\cap\mathrm{Sp}\,\psi\neq\emptyset$.

(iii) For any equality-free first order sentence ψ, if $\mathrm{Sp}\,\varphi\neq\emptyset$ and $\mathrm{Sp}\,\psi\neq\emptyset$, then either $\mathrm{Sp}\,\varphi\subset\mathrm{Sp}\,\psi$ or $\mathrm{Sp}\,\psi\subset\mathrm{Sp}\,\varphi$.

Proof. (i) is immediate by Corollary 2. (ii) and (iii) are entailed by (i).

Q.E.D.

According to this corollary, the answer for FSP 1 is in the negative for any equality-free φ if we restrict ourselves to searching for equality-free ψ, only. For FSP 1, however, we also have the following positive result.

Corollary 5. Let φ be equality-free. Then there exists a first order sentence ψ such that for any $n\in[\omega]$, $n\in\mathrm{Sp}\,\psi$ iff $n\notin\mathrm{Sp}\,\varphi$.

Proof. If $\mathrm{Sp}\,\varphi=\emptyset$, then we may choose $\psi=(\forall x)\ (x=x)$. Obviously, $\mathrm{Sp}\,\psi=[\omega]$. If $\mathrm{Sp}\,\varphi\neq\emptyset$, then, by Corollary 4 (i), there exists a least number $n_0\in[\omega]$ such that

$$Sp\,\varphi = \{n|n\in\omega \quad \text{and} \quad n\geqq n_0\}.$$

Then, $[\omega]-\mathrm{Sp}\,\varphi=\{1, 2, \ldots, n_0-1\}$ and we may choose for ψ the sentence $(\exists v_1 \ldots \exists v_{n_0-1}\forall v)\ (v=v_1 \ldots v=v_{n_0-1})$.

Q.E.D.

Let $N\subset\omega$. We say that N is a final segment of ω iff there exists an $n_0\in[\omega]$ such that $N=\{n|n\in\omega \text{ and } n\geqq n_0\}$.

For FSP 2, we have by Corollary 4, that, given the equality-free sentence φ arbitrarily, the finite spectrum of φ is a final segment of ω or else $\mathrm{Sp}\,\varphi=\emptyset$.

For the remaining two questions FSP 3 and FSP 4, we obtain:

Corollary 6. Let $N\subset[\omega]$, $N\neq\emptyset$. Then, the following two assertions are equivalent:

(i) N is the finite spectrum of an equality-free first order sentence.

(ii) N is a final segment of ω.

Proof. (i)$\Rightarrow$(ii) is immediate from Corollary 4 by definition.

(ii)$\Rightarrow$(i) Let N be a final segment of ω, $N\neq\emptyset$. Then there is a number $n_0\in[\omega]$ such that $N=\{n|n\in\omega \text{ and } n\geqq n_0\}$. Consider the set T_{n_0} for n_0 constructed in the proof of Theorem 3 (i). Obviously, T_{n_0} is finite, hence $\varphi=\wedge T_{n_0}$ is an equality-free sentence. By Theorem 3 (i), $\mathrm{Sp}\,\varphi=N$.

Q.E.D.

References

[1] ANDRÉKA, H., I. NÉMETI, The generalized completeness of Horn predicate logic as a programming language, *Acta Cybernetica,* Tom. 4., Fasc. 1., pp. 3—13.

[2] ASSER, G., Das Repräsentantenproblem im Prädikatenkalkül der ersten Stufe mit Identität, *Zeitschrift für Math. Logik und Grundlagen der Math.* Vol. 1. (1955) pp. 252—263.

[3] CHANG, C. C., H. J. KEISLER, *Model Theory* (Studies in Logic, Vol. 73) North-Holland Publ. Co., Amsterdam (1973).

[4] ECSEDI-TÓTH, P., On the expressive power of equality-free languages, in *Zeitschrift für Math. Logik und Grundlagen der Math.* Vol. 32.

[5] FAGIN, R., Generalized first order spectra and polynomial time recognizable sets, In: *Complexity of Computation* (ed. R. Karp), SIAM—AMS Proc. Vol. 7, (1974).

[6] GÁCS, P., L. LOVÁSZ, Some remarks on generalized spectra, *Zeitschrift für Math. Logik und Grundlagen der Math.* Vol. 23. (1977), pp. 547—554.

[7] GRÄTZER, G., *Universal Algebra,* Springer Verlag, New York (2nd ed.) (1979).

[8] MOSTOWSKI, A., Concerning a problem of H. Scholz, *Zeitschrift für Math. Logik und Grundlagen der Math.* Vol. 2. (1956), pp. 210—214.

[9] SCHOLZ, H., *J. Symb. Log.* Vol. 17. (1952), p. 160.

Received Febr. 9, 1984

DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1.
SZEGED, HUNGARY
H—6720

The analysis of signal flow graph containing sampled-data elements

IMRE PÁVÓ

Summary

In this paper the author reduces the input-output analysis of a signal flow graph containing sampled-data elements to the analysis of a signal flow graph consisting purely of basic (linear) elements. By this reduction, the algebraic formula of the output signals known from the literature can be substantially simplified. The new formula can advantageously be used to calculate the response signals by computer as well as by topological methods.

Introduction

The input-output analysis of a linear system excited by continuous and sampled-data signals is possible by calculating the output signals of a signal flow graph consisting of basic and sampled-data elements [1]. In practice, the methods for these calculations adopt algebraic [5] or topological apparatus [4]. The main advantage of a topological apparatus is in its graphic quality, but the application is recommendable in special cases only, for it is complicated in respect of computer technique. The algebraic method presented in [5] can be used in more general cases but the formula of the output signals is still complicated.

In this paper, a procedure is introduced which gives a simpler algebraic formula of the analysis. Hence, on the one hand, the earlier method [5] can be reduced from the point of view of computer implementation, on the other hand an effective topological procedure can be designed for more general cases. The present procedure is applicable to signal flow graphs containing sampled-data elements working synchronously, however there is no limitation either for the number of the elements or for the number of the input-output vertices.

The new algebraic formula of the analysis

Consider the signal flow graph G containing sampled-data elements of number r. Let the number of input vertices be m and that of the output vertices n. Associate with each edge of G a transfer function as a parameter, and excite the system at the input vertices by the vector $X=(x_1, ..., x_m)$ the components of which are Laplace transforms of the exciting signals. Due to this excitation, the response vector $Y=(y_1, ..., y_n)$ appears at the output of G, the components of which are Laplace transforms of the output (response) signals. For example Fig. 1 shows a signal flow graph with one input and one output and with two sampled-data elements. On Fig. 1 the sampled-data elements are marked by dotted lines and the transfer directions are also indicated by arrows. The task is to calculate the output vector Y.

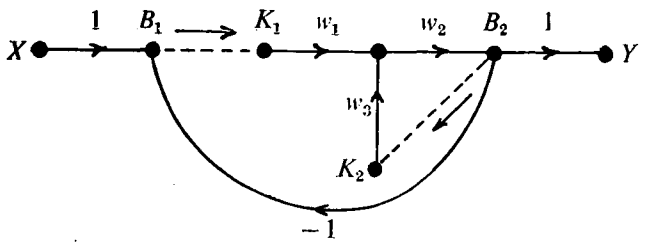


Fig. 1.

Let B be the vector of (the Laplace transform of) the signals appearing at the starting points of the sampled-data elements and denote by K the vector the components of which are (the Laplace transform of) the signals at the endpoints of the sampled-data elements.

Vector B and K are of size r.

For the calculation, we transform G as follows. Delete all sampled-data elements from G and consider $\begin{bmatrix} X \\ K \end{bmatrix}$ as the exciting vector of the remaining graph and let $\begin{bmatrix} B \\ Y \end{bmatrix}$ be the response vector. As the remaining graph is linear we can write:

$$\begin{bmatrix} B \\ Y \end{bmatrix} = \mathbf{W} \begin{bmatrix} X \\ K \end{bmatrix} \tag{1}$$

where $\mathbf{W}$ denotes the transfer matrix of the remaining graph. Moreover, the condition

$$K = B^* \tag{2}$$

must be fulfilled. The star in (2) refers to the sampling operation. During the transformation neither the topology nor the signals of the original graph change, particularly the vector Y remains the same.

Fig. 2 derived from Fig. 1 illustrates the transformation. Observe that now the input and output vectors of the remaining graph are of size 3. After partitioning $\mathbf{W}$,

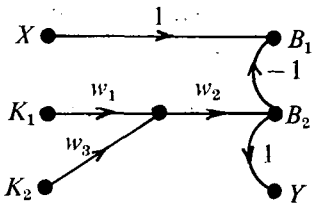


Fig. 2.

(1) can be written in the following form:

$$\begin{bmatrix} B \\ Y \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} \\ \mathbf{W}_{21} & \mathbf{W}_{22} \end{bmatrix} \begin{bmatrix} X \\ K \end{bmatrix},$$

and hence the system of equations

$$\begin{cases} B = \mathbf{W}_{11} X + \mathbf{W}_{12} K \\ Y = \mathbf{W}_{21} X + \mathbf{W}_{22} K \end{cases} \tag{3}$$

follows.

Let us consider the sampled form of the second equation of (3). Taking into account (2), we can write:

$$B^* = (\mathbf{W}_{11} X)^* + \mathbf{W}_{12}^* B^*. \tag{4}$$

Hence we obtain:

$$(\mathbf{1} - \mathbf{W}_{12}^*) B^* = (\mathbf{W}_{11} \cdot X)^*, \tag{5}$$

where $\mathbf{1}$ is the unit matrix of size $r \times r$. If $\det(\mathbf{1} - \mathbf{W}_{12}^*) \neq 0$, from (5)

$$B^* = (\mathbf{1} - \mathbf{W}_{12}^*)^{-1} \cdot (\mathbf{W}_{11} X)^* \tag{6}$$

follows, where the upper index -1 refers to the inverse matrix. Finally, substituting the right-hand side of formula (6) into the second equation of (3), and taking (2) into consideration, we have:

$$Y = \mathbf{W}_{21} X + \mathbf{W}_{22} (1 - \mathbf{W}_{12}^*)^{-1} \cdot (\mathbf{W}_{11} \cdot X)^*. \tag{7}$$

Notice that the practical application of the formula (7) requires the calculation of the transfer matrix of a linear signal flow graph, which is possible by the method elaborated in the reference [5].

Computer implementation

Comparing the response vector (7) with the transfer matrix formula of a signal flow graph consisting of basic elements only given in [5] it can be observed that both of these calculations require the same matrix operations (i.e. partitioning, substraction from the unit matrix, calculation of the inverse, multiplication). Since the application of (7) also requires the calculation of the transfer matrix of a signal flow graph with basic elements, there is a possibility to construct a common program

for the analysis of signal flow graphs without as well as with sampled-data elements. Such a program gives either the transfer matrix (first case) or the output vector (second case). Fig. 3 shows the scheme of the program mentioned above.

The input data of the program are as follows: m and n denote the numbers of the input and output vertices, r is the number of the sampled-data elements while v stands for the number of the internal vertices. The basic elements are given by

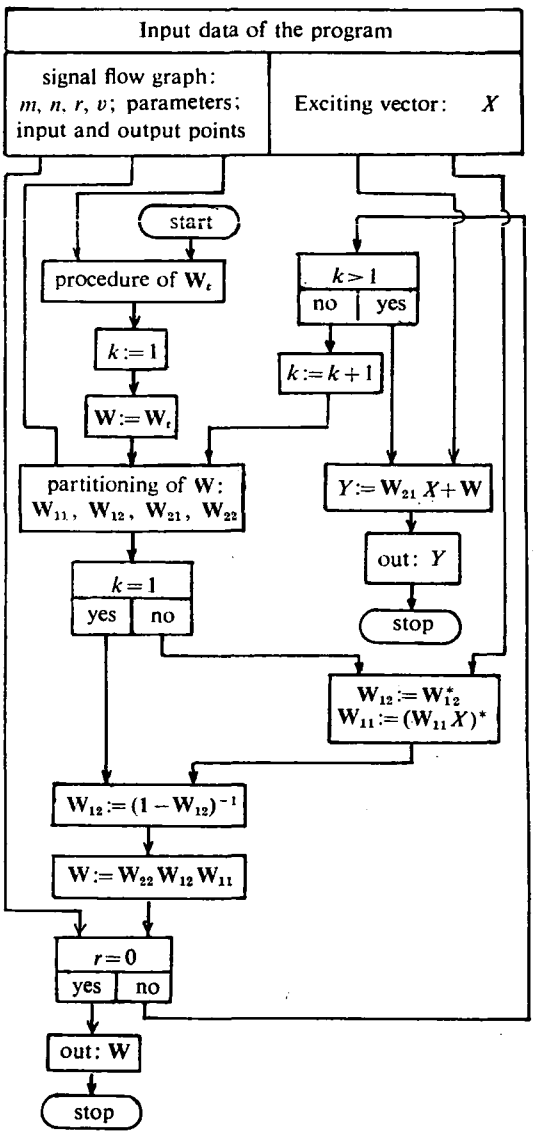


Fig. 3.

their endpoints. For the calculation of the transfer matrix, the components of vector X can be chosen arbitrarily. The program is executed in one cycle if the task is to determine the transfer matrix, or in two cycles if we wish to calculate the output vector Y. The program parameter k counts the necessary cycles and $\mathbf{W}_t$ denotes the node matrix of the original signal flow graph.

For the sake of comprehensibility we summarize the calculation of the transfer matrix on the left side of Fig. 3 (case $r=0$). If sampled-data elements are also present ($r\neq 0$), the calculation proceeds on the right-hand side of the scheme and the second cycle starts. For performing the iterative steps, the program returns to the appropriate blocks on the left-hand side of Fig. 3. The location of the necessary sampling operations can also be easily seen in the scheme.

A topological procedure

Notice that matrices $\mathbf{W}_{11}, \ldots, \mathbf{W}_{22}^{*}$ occuring in (3) can be obtained by topological formulas from the signal flow graph without sampled-data elements, namely each of them can also be regarded as a transfer matrix belonging to a special excitation. (For example to determine $\mathbf{W}_{11}$ let the input vector be $\begin{bmatrix} X \\ O \end{bmatrix}$, the output one B'; in case of $\mathbf{W}_{12}$ the input vector $\begin{bmatrix} O \\ K \end{bmatrix}$, the output one B''; because of the linearity $B=B'+B''$, and so on). For determining Y by topological method it is enough to show that K can also be produced by a topological formula.

For this purpose, let us introduce the notations $S=(\mathrm{S}_1, \ldots, \mathrm{S}_r)=(\mathbf{W}_{11}X)^*$. Taking (2) into account, we have:

$$K = S+\mathbf{W}_{12}^{*}K \tag{8}$$

Now, let us consider the signal flow graph $\mathrm{G}_{\mathbf{M}}$ associated with the linear system of equations (8) in the usual manner (MASON graph, [2]). In the general case,

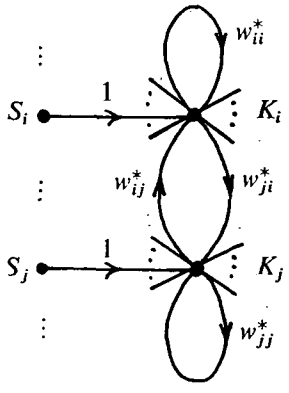


Fig. 4.

G_M arises from the directed full graph with r vertices, the edges of which are parametrized with the elements of $\mathbf{W}_{12}^*$, namely using the notation $\mathbf{W}_{12}=(w_{ij})_{r\times r}$ the parameter of the loop coinciding to the i-th vertex is w_{ii}^*, while the parameter of the edge directed from the i-th vertex to the j-th vertex is w_{ji}^* $(i,j=1,\ldots,r)$. This full graph has to be supplemented by r edges, each of which has a parameter 1 as transmission. The starting points of the supplementary edges are the inputs of G_M, the endpoints are the vertices of the full graph which are at the same time the outputs of G_M. Exciting G_M by S at the inputs, the vector K appears as the response vector in the outputs.

Fig. 4 indicates a part of G_M in a general case. From G_M the vector K can be obtained by a topological formula, and finally, the first equation of (3) gives the vector Y.

Application

Let us consider the signal flow graph with sampled-data elements given in Fig. 1 and let our first task be the calculation of the response vector Y by formula (7).

Investigating Fig. 2 one can write the vector equation (1) in the following form:

$$\begin{bmatrix} B_1 \\ B_2 \\ Y \end{bmatrix} = \mathbf{W} \begin{bmatrix} X \\ K_1 \\ K_2 \end{bmatrix} \tag{9}$$

Using the method described in [5] the transfer matrix is:

$$\mathbf{W} = \left[\begin{array}{c:cc} 1 & -w_1w_2 & -w_3w_2 \\ 0 & w_1w_2 & w_3w_2 \\ \hdashline 0 & w_1w_2 & w_3w_2 \end{array}\right] \tag{10}$$

In (10) the dotted lines indicate the partitioning of $\mathbf{W}$. After some calculation

$$\mathbf{1}-\mathbf{W}_{12}^* = \begin{bmatrix} 1+(w_1w_2)^* & (w_3w_2)^* \\ -(w_1w_2)^* & 1-(w_3w_2)^* \end{bmatrix}$$

arises. Finally we get:

$$(\mathbf{1}-\mathbf{W}_{12}^*)^{-1} = \frac{1}{1-(w_3w_2)^*+(w_1w_2)^*} \begin{bmatrix} 1-(w_3w_2)^* & -(w_3w_2)^* \\ (w_1w_2)^* & 1+(w_1w_2)^* \end{bmatrix} \tag{11}$$

Taking into account (10) and (11), from (7) we can write for the output vector:

$$Y = [w_1w_2 \quad w_3w_2] \cdot \frac{1}{1-(w_3w_2)^*+(w_1w_2)^*} \begin{bmatrix} 1-(w_3w_2)^* & -(w_3w_2)^* \\ (w_1w_2)^* & 1+(w_1w_2)^* \end{bmatrix} \begin{bmatrix} X^* \\ 0 \end{bmatrix} =$$

$$= \frac{w_1w_2-w_1w_2(w_3w_2)^*+w_3w_2(w_1w_2)^*}{1-(w_3w_2)^*+(w_1w_2)^*} \cdot X^* \tag{12}$$

As a second task let us determine Y in the previous example by topological procedure. Using the MASON formula, from Fig. 2 the matrices $\mathbf{W}_{11},\ldots,\mathbf{W}_{22}$ can immediately be read. Then it is sufficient to determine K by topological formula.

Taking into account the elements of $\mathbf{W}_{12}^*$, the signal flow graph G_M associated

with the system of equations (8) can be given. G_M is drawn on Fig. 5. Applying the MASON formula directly to Fig. 5

$$K_1 = \frac{X^*(1-(w_3w_2)^*)}{1+(w_1w_2)^*-(w_3w_2)^*}, \quad \text{and}$$

$$K_2 = \frac{X^*(w_1w_2)^*}{1+(w_1w_2)^*-(w_3w_2)^*} \tag{13}$$

are fulfilled.

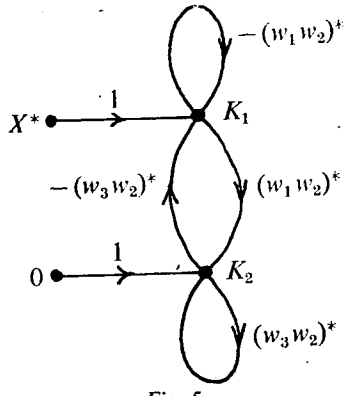


Fig. 5.

Finally, by (10) and (13) we get from the first equation of (3)

$$Y = [w_1w_2 \quad w_3w_2] \cdot \begin{bmatrix} \dfrac{X^*(1-(w_3w_2)^*)}{1+(w_1w_2)^*-(w_3w_2)^*} \\ \dfrac{X^*(w_1w_2)^*}{1+(w_1w_2)^*-(w_3w_2)^*} \end{bmatrix} \tag{14}$$

After calculating the prescribed operations in (14) the result is identical with (12).

References

[1] ASH, G. R. H., W. R. KIM, G. M. KRANC, 'A general flow graph technique for the solution of multiloop sampled systems', *Trans. of ASME, Journal of Basic Engineering*, 360—370 (1960).
[2] MASON, S. J., 'Feedback theory — some properties of signal flow-graphs', Proc. *IRE*, 41, 1144—1156 (1953).
[3] MASON, S. J., 'Feedback theory — further properties of signal graph', Proc. *IRE*, 44, 920—926 (1956).
[4] MAYEDA, W., *Graph theory*, Wiley, New York, 1972.
[5] VÁGÓ, I., 'The calculation of transfer matrices of linear systems with continuous and sampled-data signals by signal flow graphs', Proc. *CT and Applications*, 11, 355—362 (1983).

Received July 16, 1984

RESEARCH GROUP ON THEORY OF AUTOMATA OF THE
HUNGARIAN ACADEMY OF SCIENCES
SOMOGYI U. 7.
SZEGED, HUNGARY
H—6720

Network design problem: structure of solutions and dominance relations

E. BURATTINI*, G. MARRA**, A. SFORZA**

Abstract

The network design problem (NDP), in its simplest form, is that of designing a connected subnetwork of an n node network, by selecting from the set of all edges a subset which minimizes the sum of user's shortest path costs between all node pairs of the networks, being subjected to a budget constraint which limits the number of edges that may be included in the optimal network.

In the present paper, remembering the complexity of the NDP (10) and the branch and backtrack (B&Bt) procedures applied to it, we point out the opportunity of reducing the total number of operations, required to solve it, using some dominance relations existing among its solutions. An algorithm which uses such relations is also proposed.

Introduction

The network design problem (*NDP*) is a well defined subject of transportation planning. In its general form it can be defined as follows: given a connected graph $G=(N, A)$ with n nodes and m edges; a subset of edges which can be invested in (improved or constructed); a set of investment costs on these edges; a set of user's costs on the edges with and without investments; a set of origin-destination (o/d) pairs on the graph; a set of demanded flows between o/d pairs: find the set of edges which minimizes the total user's cost with a budget constraint on the total investment cost. This problem is interesting since its solution may be relevant in the design of transportation networks. In all these applications the network design is obviously subjected to many more constraints than those considered in this paper,

* Istituto di Cibernetica — C.N.R. Via Toiano, 6 Arco Felice, Napoli, Italy.
** Facoltà di Ingegneria di Napoli, Cattedra di Ricerca Operativa Via Claudio, 21, Napoli, Italy.

but the solution of the NDP may be used as a measuring standard for the efficiency of applicative designs, and this justifies the study of the NDP.

Under the hypothesis that the demand flows are equal to 1 for all the *o/d* pairs on the graphs and that the subset of edges which can be invested in is equal to the set of all edges, it is possible to define a simplified version of the network design problem, i.e. to find the set of edges which minimizes the sum of the shortest path costs between all pairs of nodes with a budget constraint on the total investment cost.

This combinatorial problem is expressed by a binary programming model whose variables are associated to the edges which can be included in the network.

It belongs to the NP-complete class and then it requires exponential computation time (10).

Branch and bound techniques are generally used for its exact or approximate solution (3, 4, 5, 6, 7). The structure of these algorithms are substantially the same. It is based on two tests, lower bound test and feasibility test, applied to the partial problem P_i, generated during the procedure of separation and progressive evaluation of the B&B.

The lower bound test generally used is not powerful enough to exploit the aspects of the problem structures which are useful to improve the computation efficiency of the algorithms. It is possible to define a dominance relation $\boldsymbol{D}$ among the subproblems P_i of the NDP such that if $P_i \boldsymbol{D} P_j$ then the objective function value $f(P_i)$ is not greater than $f(P_j)$. If it occurs, and P_i has already been evaluated, then we can exclude from consideration the subproblem P_j.

The use of suitable parameters associated to the nodes of the B&B arborescence, allows us to evaluate the "goodness" of a solution and to define a bounding strategy useful to reduce the objective function evaluations number.

An algorithm which uses the dominance relation and this bounding strategy is proposed.

Mathematical formulation

Let $G=(N, A)$ be a connected undirected graph, with

$N \equiv \{v_1, \ldots, v_n\}$	set of nodes,
$A \equiv \{\alpha_1, \ldots, \alpha_m\}$	set of all possible edges,
$T \equiv \{\langle \omega, \delta \rangle \in N \times N,\ \omega \neq \delta\}$	set of origin-destination (o/d) pairs.

We define the following functions:

$C\colon A \to R^+;\ C \equiv \{c_1, \ldots, c_m\}$	set of user's costs on the edges,
$H\colon P \to R^+;\ H \equiv \{h_1, \ldots, h_m\}$	set of the investment costs on the edges.

Let $B \in R^+$ be the budget.

We can state the following problem:

Find a subgraph

$$G' = (N, A') \quad (A' \subseteq A)$$

such that

$$U(X) = \sum_{\langle \omega, \delta \rangle \in T} l_{\omega,\delta}(X) \text{ min!}$$

$$\sum_{i=1}^{m} x_i h_i \leqq B$$

where $l_{\omega,\delta}(X)$ is the shortest path cost between ω and δ with $X \equiv \{x_1, \ldots, x_m\}$ i.e.:

$$x_i = \begin{cases} 1 \text{ if the edge } i \text{ is included in the network,} \\ 0 \text{ otherwise,} \end{cases}$$

and

$$l_{\omega,\delta}(X) = \sum_{i: a_i \in \varrho} c_i x_i$$

where ϱ is the shortest path between ω and δ.

This problem was proved by Johnson et al. (10) to be NP-complete and by Wong (15) to be NP-hard i.e. it is quite unlikely to find an algorithm for it such that its running time be a polynomial function of the input size. Then, if we want to solve such a problem in an optimal way using the classical B&Bt techniques, it may happen that it would take too much time. For this reason, it is useful to apply branch and backtrack algorithms with some functions able to reduce at the minimum the branching on the B&Bt arborescence.

Branch and Backtrack Algorithm and Dominance Relations

The B&Bt is a computational principle which has been proved useful in solving various combinatorial optimization problems encountered in operations research and combinatorial mathematics. The underlying idea of a B&Bt procedure is to decompose a given problem into smaller and smaller partial problem. Two types of tests are applied to each partial problem to see if it can be solved or, on the contrary, be concluded that no optimal solution is obtainable from it; in both cases, the partial problem is terminated and not decomposed any further. These tests are called lower bound (for minimization problem) and feasibility tests. The computation terminates when all nodes are either decomposed or terminated. In the NDP the feasibility test is done verifying if the best solution of the subproblem P_i satisfy the constraint of the budget B. The lower bound $g(P_i)$ of the optimal value $f(P_i)$ of a partial problem P_i is generally found evaluating the objective function value of the best solution of P_i, obtained setting to 1 the free variables of P_i.

If $g(P_i) \geqq z$ where z is the current optimum, that is the value of the best feasible solution of P_0 (the given minimization problem) obtained so far, we conclude that P_i does not provide an optimal solution of P_0, and P_i is terminated. A generalization of this lower bound test can be done using a binary relation $\boldsymbol{D}$, called dominance relation.

Let $\boldsymbol{P} \equiv \{P_1, \ldots, P_p\}$ be the set of partial problems generated by a B&Bt procedure. A dominance relation $\boldsymbol{D}$ is a partial order relation over $\boldsymbol{P}$ which satisfies the following conditions (7).

1) $\boldsymbol{D}$ is:
   transitive $P_i \boldsymbol{D} P_j \ \& \ P_j \boldsymbol{D} P_k \rightarrow P_i \boldsymbol{D} P_k$,
   reflexive $P_i \boldsymbol{D} P_i$,
   antisymmetric $P_i \boldsymbol{D} P_j \ \& \ P_j \boldsymbol{D} P_i \rightarrow P_i = P_j$;

2) $P_i \mathbf{D} P_j \;\&\; P_i \neq P_j \rightarrow f(P_i) \geqq f(P_j)$ and
P_i is not a proper descendant of P_j;

3) $P_i \mathbf{D} P_j \;\&\; P_i \neq P_j$ imply that some descendant
$P_{i'}$ of P_i satisfies $P_{i'} \mathbf{D} P_{j'} \;\&\; P_{i'} \neq P_{j'}$ for any descendant $P_{j'}$ of P_j.

It is obvious that P_j need to be solved if P_i is already generated and $P_i \mathbf{D} P_j$ holds; thus P_j can be terminated. A dominance relation $\mathbf{D}$ may be interpreted as an embodiment of the information on optimal solutions of partial problems obtainable without actually solving them (that is computing $f(P_j)$), and can be regarded as a generalization of the lower bound test.

For the NDP we can assert that the total cost function $U(X)$ is a monotone non-increasing function of the decisional variables $\{x_i\}$. Consider two solutions X^J and X^K, best solution of the partial problems P_j and P_k.

We say that

$$X^J \geqq X^K \quad \text{if} \quad \forall i x_i^J \geqq x_i^K.$$

The following statements hold:

i) $X^J \geqq X^K \;\&\; X^K \geqq X^M \rightarrow X^J \geqq X^M$,

$X^K \geqq X^K$,

$X^J \geqq X^K \;\&\; X^K \geqq X^J \rightarrow X^K = X^J$,

ii) $X^J \geqq X^K \rightarrow U(X^J) \leqq U(X^K)$.

Consider an edge i such that $x_i^J \geqq x_i^K$ (that is $x_i^J = 1$ and $x_i^K = 0$);

the flow unit between each pair $\langle \omega, \delta \rangle$ using through X^J a shortest path containing α_i, will use through X^K another path of cost $l_{\omega,\delta}(X^K) \geqq l_{\omega,\delta}(X^J)$,

iii) $X^J \geqq X^K \;\&\; X^J \neq X^K \rightarrow$

$\forall K' \colon X^{K'} \leqq X^K \exists J' \colon X^{J'} \leqq X^J \;\&\; X^{J'} \geqq X^{K'}$.

If we consider the solutions X^J and X^K as best solutions of the problems P_j and P_k we can state that

$$X^J \geqq X^K \leftrightarrow P_j \mathbf{D} P_k. \tag{a}$$

In Fig. 1 the complete arborescence of an NDP with 4 variables is represented. In Fig. 2 a sequential graph which takes care of all dominance relations among the solution is shown. To each level of this graph the solutions with the same number of variables set to 1 belong. Generally, the known B&B algorithms use only partially the dominance relation. The thick line represents the dominance relations implicitly considered in B&Bt algorithms, the sharp line, the other dominance relations.

A more readable version of the graph of Fig. 2 is reported in Fig. 3. It can be remarked that:

a) none of the solution is dominated by any other of the successive level,
b) none of the solutions dominated by at least one solution belonging to the previous level,

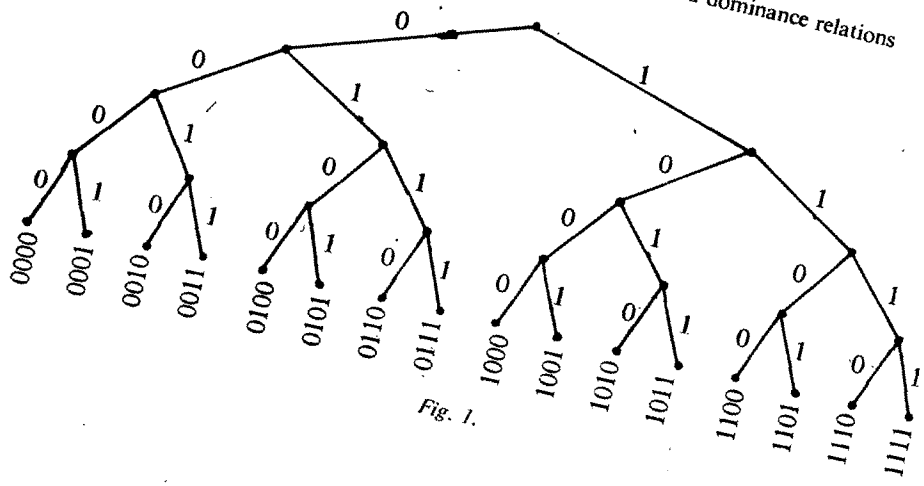


Fig. 1.

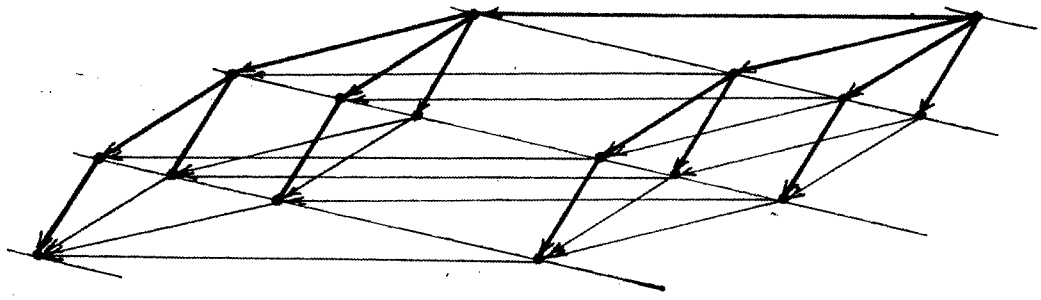


Fig. 2.

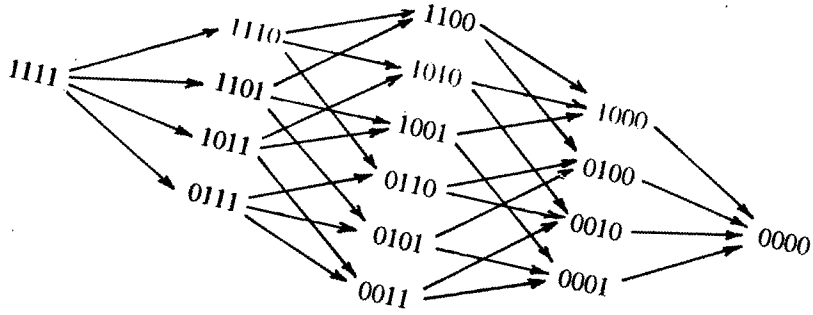


Fig. 3.

c) solutions belonging to the same level are not comparable using the $\boldsymbol{D}$ relations.

Obviously the number of level is equal to $m+1$ and for each level there are $\binom{m}{l}$ different solutions, ($l=0, 1, \ldots, m$).

If a problem P_i is dominated by another P_j already solved with $U(X^J) \geqq z$ (current upper bound) we do not need to solve P_i, that is to compute the relative objective function value.

We underline that applying the dominance relation does not reduce the number of the generated partial problems but only the number of the solved problems.

Some remarks on the structure of the solutions

The graphical representation of Fig. 3 enables us to identify quickly the subsets of feasible and infeasible solutions. In fact the "border" between these two subsets can be defined as the subset of those feasible solutions, dominated by the infeasible ones.

If the investment costs h_i ($i=1, \ldots, m$) are all equal to $\bar{h}$ the border is represented by the level $[B/\bar{h}]$. If the h_i costs are not equal, the border can be represented by solutions belonging to different levels. In this case, arranging the h_i in non-decreasing order, the level l satisfying the conditions

$$\sum_{i=1}^{m+1-l} h_i \leqq B \quad \text{and} \quad \sum_{i=1}^{m+2-l} h_i \geqq B$$

is the first level with border solutions, that is the level with border solutions having the highest number of 1. The remaining feasible solutions of the border belong to successive levels ($l+1, \ldots, m$) and can be identified starting from the infeasible solutions of level l.

The optimal solutions of the problem belong to the border, on which all solutions are not comparable and altogether dominate all the other feasible ones.

Using the dominance relation without solving the problems P_i, whose best solutions are infeasible, we must solve all and only those belonging to the border. If we do not use the dominance relation, the classical criteria applied in B&Bt algorithm does not prevent us from analyzing solutions not belonging to the border and then obviously not optimal.

The proposed algorithm

The algorithm we propose here is a classical B&Bt algorithm with the addition of the dominance test and of some heuristic devices for the bounding strategy. A more general scheme of our algorithm is reported in Fig. 4.

Some detailed steps are the following

— The algorithm realizes a preliminary arrangement of the variables to speed-up the procedure.

— A feasibility test verifies if the available budget allows us to set up an other variable to 1.

```
      begin
1 —       𝒜={P0}; z←∞; 𝒪←Φ
      end
      begin
2 —       while 𝒜≠Φ do
3 —       Pi=s(𝒜)
4 —       if Pj𝒟Pi for some Pj≠Pi belonging to the set of nodes currently generated then
              begin
5 —               𝒜=𝒜−{Pi}
              end
          else
6 —           if Pi can be solved or proved to be infeasible then
                  begin
7 —                   if z>f(Pi) then
                          begin
8 —                           𝒪=0(Pi); z=f(Pi)
                              go to 11
                          end
                      else
9 —                       if z=f(Pi) then
                              begin
10 —                              𝒪=𝒪∪0(Pi)
                              end
11 —                  𝒜=𝒜−{Pi}
                  end
              else
12 —              if g(Pi)>z then
                      begin
13 —                      𝒜=𝒜−{Pi}
                      end
                  else
14 —                  generate sons Pi1, ..., Pik of Pi; 𝒜←𝒜∪{Pi1, ..., Pik}−{Pi}
      end
      begin
15 —      if z=∞ then
              begin
16 —              print "P0 is infeasible"
              end
          else
17 —          0(P0)←𝒪; f(P0)←z
      end
```

Fig. 4.

— A first test on budget left compares the minimum value of the free variables investment costs with the remaining available budget.
— The dominance test compare the best solution of the current subproblem with the list of the not-yet dominated solution.
— A second test on budget left compares the sum of the free variables investment costs with the remaining available budget.
— A solution "goodness" test compares the value of a solution "parameter" with the parameter value of the current upper bound solution.

The algorithm has been tested on small dimension networks (6 nodes, 13 edges) varying user's and investment costs.

The results showed that the dominance relation effect is relevant when many $U(X)$ values must be evaluated, that is when the budget is approximately the 50%

of the $\sum_{h_i \in H} h_i$. It is useful to define some conditions about the number of requested operations.

Let T_A be the number of solved problems, t_A the operation number to solve each partial problem, t_D the average operation number for the dominance test, T_D the number of generated and dominated problems. With good approximation we can state the total operations number T', using the dominance test, is

$$T' = t_D(T_A+T_D)+T_A t_A$$

and the total operations number T' without using the dominance test, is

$$T'' = (T_A+T_D)t_A.$$

Introducing dominance test is useful until $T'<T''$, i.e.

$$T_D > \frac{t_D}{t_A}(T_A+T_D).$$

Finally, it seems useful to introduce the dominance relation in B&Bt procedure for NDP, also if the running time to conclude the arborescence remain relevant in some cases. The optimal solution, with a good arrangement of the variables is fastly found, before the end of the procedure. This considerations allows us to use the algorithm as heuristic, stopping it before the end, and getting the last feasible solution (current optimum).

Conclusions

In this paper we developed some considerations on the structure of the solution set of the problem, identifying a border of feasible solutions between the two subsets of feasible and infeasible solutions.

We have also shown the opportunity of introducing a dominance test, regarded as a generalization of the lower bound test, in the basic structure of the B&Bt algorithm for the NDP.

The principal effect of this introduction is to improve computational efficiency of the B&Bt procedure. In fact in this way a larger number of partial problems can be terminated without evaluating the objective function value.

The first results suggested the fitness of using the dominance test in a quite well defined range of the investment costs. Moreover the considerations on the solutions set structure, suggested to construct an "ad hoc" algorithm which examines directly the set of feasible solutions of the border and the set of the infeasible solutions which dominate the border.

References

[1] Beale, E. M. L., M. G. Kendall, D. W. Mann (1967), The discarding of variables in multivariate analysis, *Biometrika*, 54, 357—366.

[2] Boyce, D. E., A. Farhi, R. Weischedel (1973), Optimal network problem: a branch and bound algorithm, *Environment and Planning*, 5, 529—533.

[3] BOYCE, D. E. eds. (1979), Transportation Research 13B, Special Issue on Network Design Problem.
[4] FLORIAN, M., R. DIONNE (1977), Exact and approximate algorithms for optimal network design, *Publication 41, Centre de recherche sur les transports,* Montreal.
[5] GALLO G. (1981), A new branch and bound algorithm for the network design problem, *"Optimization Days"* Montreal.
[6] GALLO, G. (1981), "Lower planes for the Network Design Problem", S—81—22 — ISI — Pisa.
[7] HOANG, H. H. (1973), A computational approach to the selection of an optimal network, *Management Science,* 19, 5, 488—498.
[8] IBRAKI, T. (1977)a, On the computational efficiency of the branch and bound algorithms, *Journal of the Operations Research of Japan,* 20, 1, 16—35.
[9] IBRAKI, T. (1977)b, The power of the dominance relation in branch and bouhd algorithm, *Journal of the Association for Computing Machinery,* 24, 2, 264—279.
[10] JOHNSON, D. S., J. K. LENSTRA, A. H. G. RINNOY KAN (1978), The complexity of the network design problem, *Networks,* 8, 279—285.
[11] LE BLANC, L. J. (1975), An algorithm for the discrete network design problem, *Transportation Science,* 9, 183—199.
[12] OCHOA-ROSSO, F., A. SILVA (1968), Optimum project addition in urban transportation network via descriptive traffic assignement models, *Research Report R68—44,* Transportation System Division, MIT, Cambridge, Massachusets.
[13] RIDLEY (1968), An investment policy to reduce the travel time in a transportation network, *Transportation Research,* 2, 409—424.
[14] STEENBRINK, P. A. (1974), *Optimization of transport network,* Wiley, New York, *Transportation Research* (1979) 13B, Special Issue on network design problem.
[15] WONG R. T. (1980), "Worst-case analysis of Network Design Problem Heuristics", Siam J. Alg. Discr. Math., Vol. 1, N. 1, March, 51—63.

Received June 2, 1983

Bibliographie

R. Gleaves: Modula-2 for Pascal Programmers, X+145 pages, Springer-Verlag, New York, 1984

"Modula-2 is a general-purpose language designed primarily for writing software systems. The name Modula is short for modular language.
Modula-2 is the third in a family of languages created by the Swiss computer scientist Niklaus Wirth. The first language, Pascal, was conceived as a teaching language, but achieved widespread use. The second, Modula, was a special-purpose language designed for programming small real-time control systems. Modula-2 emerged as a synthesis of the systems programming capabilities of Modula and the general utility of Pascal."

This book is written for people who know the Pascal language and who wish to learn Modula-2 in terms of their knowledge of Pascal. The book does not offer a complete description of the language; it is intended to give an overview of the language by focussing on the differences from Pascal and by introducing new concepts unique to Modula-2. A major strength of the book lies in its practical approach. The book is divided into three parts and the appendices. The appendices include syntax diagrams and a glossary of Modula-2 terminology. Part 1 introduces language concepts which are unique to Modula-2. These are modules, separately compiled modules, program and subprogram modules, utility modules, module library, low-level programming coroutines and interrupts. Numerous example programs are provided to illustrate the new language concepts. Part 2 discusses the differences between Pascal and Modula-2, Part 3 presents a set of utility modules which are not part of the Modula-2 language. These modules provide the basic programming facilities used by most Modula-2 programs and are part of an existing Modula-2 system.

People who are familiar with Pascal can use this book as a good overview of the language Modula-2.

Gy. Horváth

Machine Learning. An Artificial Intelligence Approach. Edited by R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Symbolic Computation) XI+572 pages, Springer-Verlag, Berlin—Heidelberg—New York—Tokyo, 1984.

We may start with a brief outline of the book.

In Chapter 1, Carbonell, Michalski and Mitchell give an overview of Machine Learning (objectives, taxonomy, historical sketch).

At the Carnegie-Mellon Machine Learning Workshop in July, 1980, Herbert Simon was asked to deliver the keynote address, where he chose to play the role of devil's advocate and ask the question "Why Should Machines Learn?" After dispelling some common myths, Simon concluded with a clarified and more appropriate set of reasons why one ought to pursue machine learning research. Chapter 2 is based almost entirely on that rather controversial keynote address.

In Chapter 3, Dietterich and Michalski analyze some well-known work in concept acquisition from a unified perspective. This part provides a general framework for the comparison of different concept-acquisition systems.

In Chapter 4, Michalski describes a general theory and methodology for the inductive learning of structural descriptions from examples. Various generalization rules are presented and discussed. The methodology developed is illustrated by a problem from the area of conceptual data analysis.

In Chapter 5, Carbonell examines the issue of learning from experience. A general planning and problem-solving paradigm is proposed, based on a computationally-effective model of analogical reasoning.

In Chapter 6, Mitchell, Utgoff and Banerji investigate the issue of acquiring and refining problem-solving heuristics by examining solutions to symbolic integration problems. Like Carbonell's approach, learning is based on past problem-solving experience, but Mitchell at al focus on acquiring heuristics for applying known strategies, rather than generalizing recurring behavior into reusable plans.

In Chapter 7, Anderson examines human problem-solving in the context of providing justifications for geometric proofs. He relies entirely upon a production system framework to encode domain knowledge, learning heuristics and problem-solving strategies.

In Chapter 8, Hayes—Roth investigates the issue of improving flawed or incomplete theories that quide plan formation in a given domain. He presents five heuristic methods and applies them to problem-solving in playing the card game hearts.

In Chapter 9, Lenat focusses on methods for learning from observation and discovery. He analyizes three domains in which heuristics plays a dominant role in guiding a search through the space of possible concepts or processes one may acquire.

In Chapter 10, Langley, Simon and Bradshaw discuss their BACON system and its application to rediscovering some basic laws of chemistry. BACON applies the principles of scientific inquiry first elucidated by Sir Francis Bacon to find the simplest numerical relations that are invariant across sets of measurements. Although not able to design its own experiments, given the unanalyzed results of appropriate chemical experiments, BACON has rediscovered such laws as Guy-Lussac's law and Proust's law of definite proportions.

In Chapter 11, Michalski and Stepp investigate the problem of the automated construction of taxonomies of observed events in a manner that is meaningful to a human. They present an algorithm that implements the "conceptual clustering" operation and demonstrate its utility for the tasks of formulating descriptions of plant diseases from obsereved symptoms and taxonomizing Spanish songs in a manner meaningful to a musicologist. In contrast with statistical clustering techniques, the conceptual clustering algorithm produces characteristic descriptions of the concepts defined by each cluster.

In Chapter 12, Mostow discusses the process of learning by taking advice. Declaratively stated advice must be transformed into operational procedures effective in a given task domain. Mostow focusses on the general issue of providing advice for a heuristic search mechanism, as applied to playing the game of hearts.

In Chapter 13, Haas and Hendrix investigate the issue of automatically extending a natural language interface by acquiring domain semantics, dictionary entries and syntactic patterns from the user. The most significant aspect of their KLAUS system is that the user need not be a computational linguist, but rather is guided by the system into providing exemplary information that is later transformed into effective grammar and dictionary representations.

In Chapter 14, Rychener provides a retrospective analysis of the instructable production system project, in which many different instructional techniques for learning by being told were tried, different organizations of the knowledge were considered, and different problem-solving strategies were investigated. He concludes his chapter with an analyzis of the organizational and instructional principles that a production-system based instructional learner should adhere to in order to maximize his chances for successful knowledge acquisition.

In Chapter 15, Quinlan presents a method for generating efficient decision trees for classifying given exemplars, and applies his method to the analysis of king-and-rook versus king-and-knight chess endgames.

In Chapter 16, Sleeman investigates the application of machine learning to inter models of students learning algebra. An interesting aspect of Sleeman's work is that the teacher, in order to be effective, must learn to adapt to the student's needs, indicating that machine learning can help to make computer-assisted human education more effective.

This is a very well-written book. The chapters have a unified style and progression. From the first few chapters the reader not familiar with this field may acquire a general understanding. The second part of the book gives an excellent state of the art of machine learning.

J. Csirik

Model Program Committee of the IEEE Computer Society, "Model Program in Computer Science and Engineering", Committee Report, VI + 154 pages, IEEE Computer Society Press, 1983.

"In late 1981 the Educational Activities Board of the IEEE Computer Society reviewed the curriculum recommendations made in 1977. This evaluation clearly indicated that major changes had taken place which were not included in the earlier curriculum recommentations and that a major revision was needed. The committee established to consider this problem not only recommended that the 1977 model curriculum report be updated, but that it be expanded to cover all areas defining an exellent-caliber undergraduate program in the computer area. Therefore, in addition to the curriculum, this report addresses guidelines for the development of faculty, administration, and material resources.

The Model Program has the following goals:

— to provide an overview of the desirable features of undergraduate academic programs in computer science and engineering;
— to provide a standard of comparison that can be used to guide the development of new programs or the modification and upgrading of established programs;
— to provide an interpretation of Accreditation Board for Engineering and Technology (ABET) criteria for minimum program standards, particularly for departments seeking ABET accreditation;
— to establish a set of standards that can be used to define 'Target of Exellence' programs;
— to define the computer science and engineering aspects of a curriculum in a way that allows flexibility to meet the requirements of individual institutions;
— to provide guidance to academic administrators concerning the level of commitment needed to support a program."

The report consists of five sections, appendices and references. Section A is the introduction. A set of program criteria specifies requirements for computer science and engineering, discussed in Section B. Section C deals with all aspects of an undergraduate program in computer science and engineering. The main part of the report is the description of those 28 subject areas that constitute the core of a computer science and engineering curriculum. Subject areas are organized as collections of modules. Each module includes:

— the specification of the teaching purpose,
— the indications of the prerequisite modules,
— the concepts relevant to the module,
— references.

From the collection of subject areas, many different curricula can be generated. Subsection C. 4 shows several curricula implementations and how subject areas map into specific courses. The subject areas are the following: lecture component: fundamentals of computing, data structures, system software and software engineering, computing languages, operating systems, logic design, digital systems design, computer architecture, interfacing and communication; laboratory component: introduction to computing laboratory, software engineering laboratory, project laboratory; advanced subject areas: software engineering, digital design automation, theory of computing, database systems, advanced computer architecture, design and analysis of algorithms, fault-tolerant computing, performance prediction and analysis, computer graphics, VLSI system design, translator writing systems, computer communications networks, system laboratory, artificial intelligence, advanced operating systems. Section D provides faculty conciderations for computer science and engineering programs. Section E discusses computing, laboratory and library resource requirements of computer science and engineering programs. There are more than 400 entries in the references.
This report is recommended for those people engaged in educational activities in computer science and engineering.

Gy. Horváth

Ada: ®Language, compilers and bibliography. Ed. by M. W. Rogers (The Ada Companion Series), Cambridge University Press, 1984.

Ada is a programming language designed in accordance with requirements defined by the United States Department of Defense. It is a modern algorithmic language with the usual control structures, and with the ability to define types and subprograms. It also serves the need for modularity, whereby data, types and subprograms can be packaged. It treats modularity in the physical sense

® Ada is a registered trademark of the U.S. Government, Ada Joint Program Office

as well, with a facility to support separate compilation. In addition to these aspects, the language covers real-time programming, with facilities to model parallel tasks and to handle exceptions. It also covers systems programming; this requires precise control over the representation of data and access to system-dependent properties. Finally, both application-level and machine-level input-output are defined.

This book is divided into two sections differing in the colour of the pages. The white section contains the reference manual for the Ada programming language (ANSI/MIL—STD—1815A—1983). Some helps are added to facilitate reading of this section, which are not parts of the standard definition: Glossary, Syntax Summary, Implementation-Dependent Characteristics and Index. For example, the Index includes an entry for each technical term or phrase that is defined in the reference manual.

Guidelines for Ada compiler specification and selection are given in the green section. The purpose of this guide is to list the characteristics of an implementation that should be taken into account in the specification or selection of an Ada compiler. The section ends with a selective bibliography for Ada, with 20 aspects, including more than 460 items. This well-structured bibliography lists the principle works on Ada, covering all aspects from the history and evolution of the language to the latest thinking on the many features combined for the first time in Ada.

This book is a member of the Ada Companion Series. "This definitive new series aims to be the guide to the Ada software industry for managers, implementors, software producers and users. It will deal with all aspects of the emerging industry: adopting an Ada strategy, conversion issues, style and portability issues, and management. To assist the organised development of an Ada-orientated software components industry, equal emphasis will be placed on all phases of life cycle support."

This is a reference volume that should never be far from the workbench of the serious software engineer or programmer using Ada.

K. Dévényi

Kurt Melhorn, Data Structures and Algorithms, Vol. 1: Sorting and Searching, XII+336 pages, Vol. 2: Graph Algorithms and NP-Completeness, XII+260 pages, Vol. 3: Multi-dimensional Searching and Computational Geometry, (EATCS Monographs on Theoretical Computer Science), Springer-Verlag, Berlin—Heidelberg—New York—Tokyo, 1984.

The rapid development of computer architectures and the ever increasing complexity of computer applications have produced a revolution in mathematics. The concept of an efficient algorithm has become the centre of investigations in a large part of computer science. These 3 volumes not only give a collection of efficient algorithms, but explain what efficiency means and what principles are used to construct and analyse efficient algorithms.

Volume 1 consists of 3 chapters. Chapter I starts by introducing a machine model of computation and complexity measures. This is followed by a discussion on basic data structures: queues, stacks, lists, etc. Chapter II gives an extensive treatment of sorting algorithms. General sorting methods include heapsort, quicksort and mergesort. Specific sorting methods are presented for sorting words and reals. Chapter III is devoted to one-dimensional searching techniques, such as digital search trees, hashing, weighted and balanced trees.

Volume 2 contains 3 chapters. In Chapter IV graph representations and various graph algorithms are dealt with. The algebraic interpretation of path problems on graphs leads to efficient matrix multiplication algorithms in Chapter V. Chapter IV provides a study of NP-completeness. After a series of well-known NP-complete problems, methods for solving them are investigated.

Volume 3 has 2 chapters. Chapter VII deals with multi-dimensional searching, and Chapter VIII explores computational problems and their solutions in geometry.

Chapter IX, which is included in all 3 volumes, gives an orthogonal overview of the main algorithmic paradigms.

The volumes are written in a readable and lucid style. Except for Chapter IX, each chapter ends with exercises and bibliographic notes.

The material is self-contained. Thus, the books can be recommended to everyone interested in the subject. A large proportion of the discussion is about very recent results, which ensures that even experts will find them interesting reading.

Z. Ésik

Interactive Programming Environments (Editors: D. R. Barstow, H. E. Shrobe, E. Sandewall), XII + 610 pages, McGraw-Hill Book Company, 1984.

This book provides a collection of selected papers on interactive programming environments. Some of the papers have been published previously. The papers are divided into five sections. "The first section includes papers which describe the motivations and characteristics of interactive programming. The second section includes papers dealing with specific environments for particular languages and situations. The third section is concerned with specific issues which arise in many different environments. The fourth section includes descriptions of experimented systems which test the role of artificial intelligence in interactive programming environments. The final section includes papers concerned with the future of interactive programming environments."

The contents of the book are the following:

Section 1: Perspectives on Interactive Programming Environments. T. Winograd: Breaking the Complexity Barrier (Again); B. A. Sheil: Power Tools for Programmers; E. Sandewall: Programming in an Interactive Environment: The Lisp Experience

Section 2: Modern Interactive Programming Environments. W. Teitelman, L. Masinter: The Interlisp Programming Environment; T. Teitelbaum, T. Reps: The Cornell Program Synthesizer: A Syntax — Directed Programming Environment; J. Wilander: An Interactive Programming System for Pascal; V. Donzeau-Gouge, G. Huet, G. Kahn, B. Lang: Programming Environments Based on Structured Editors: The MENTOR Experience; A. Goldberg: The Influence of an Object-Oriented Language on the Programming Environment; B. W. Kernighan, J. R. Mashey: The UNIX Programming Environment; T. Cheatham, J. Townley, G. Holloway: A System for Program Refinement

Section 3: Aspects of Interactive Programming Environments. W. J. Hansen: User Engineering Principles for Interactive Systems; W. Teitelman: Automated Programmering: The Programmer's Assistant; W. Teitelman: A Display-Oriented Programmer's Assistant D. R. Barstow: A Display-Oriented Editor for Interlisp; R. M. Stallman: EMACS: The Extensible, Customizable, Self-Documenting Display Editor; R. D. Greenblatt, T. F. Knight, Jr., J. Holloway, D. A. Moon, D. L. Weinreb: The LISP Machine; T. A. Dolotta, R. C. Haight, J. R. Mashey: UNIX Time-sharing System: The Programmer's Workbench; A. I. Wasserman: Software Tools in the User's Software Engineering Environment; I. P. Goldstein, D. G. Bobrow: A Layered Approach to Software Design; J. W. Goodwin: Why Programming Environments Need Dynamic Data Types; E. Sandewall, C. Strömberg, H. Sörensen: Software Architecture Based on Communicating Residential Environments.

Section 4: Artificial Intelligence in Interactive Programming Environments.
C. Rich, H. E. Shrobe: Initial Report on a Lisp Programmer's Apprentice; R. C. Waters: The Programmer's Apprentice: Knowledge Based Program Editing; E. Kant, D. R. Barstow: The Refinement Paradigm: The Interaction of Coding and Efficiency Knowledge in Program Synthesis

Section 5: The Future of Interactive Programming Environments.
T. Winograd: Beyond Programming Languages; J. N. Buxton, L. E. Druffel: Rationale for Stoneman; S. E. Fahlman, S. P. Harbison: The Spice Project; D. E. Barstow, H. E. Shrobe: From Interactive to Intelligent Programming Environments

The book covers a broad field of interactive programming environments. The focus is mainly on interactive programming environments, and the related traditional areas of computer sciences are also discussed (software engineering, programming languages, artificial intelligence). The book presents a clear picture of the experiments performed in the 1970s, and all people working in this fiedl may benefit from these papers.

J. Csirik

A SZERKESZTŐ BIZOTTSÁG CÍME:

6720 SZEGED
SOMOGYI U. 7.

EDITORIAL OFFICE:

6720 SZEGED
SOMOGYI U. 7.
HUNGARY

Information for authors

Acta Cybernetica publishes only original papers in the field of computer sciences mainly in English, but also in French, German or Russian. Authors should submit two copies of manuscripts to the Editorial Board. The manuscript must be typed double-spaced on one side of the paper only. Footnotes should be avoided and the number of figures should be as small as possible. For the form of references, see one of the articles previously published in the journal. A list of special symbols used in the manuscript should be supplied by the authors.

A galley proof will be sent to the authors. The first-named author will receive 50 reprints free of charge.

INDEX — TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1984. december 19.
Megjelenés: 1984. július
Terjedelem: 8,4 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 5601 és az MSZ 5602—55 szabvány szerint
84-5165 — Szegedi Nyomda — F.v.: Dobó József igazgató